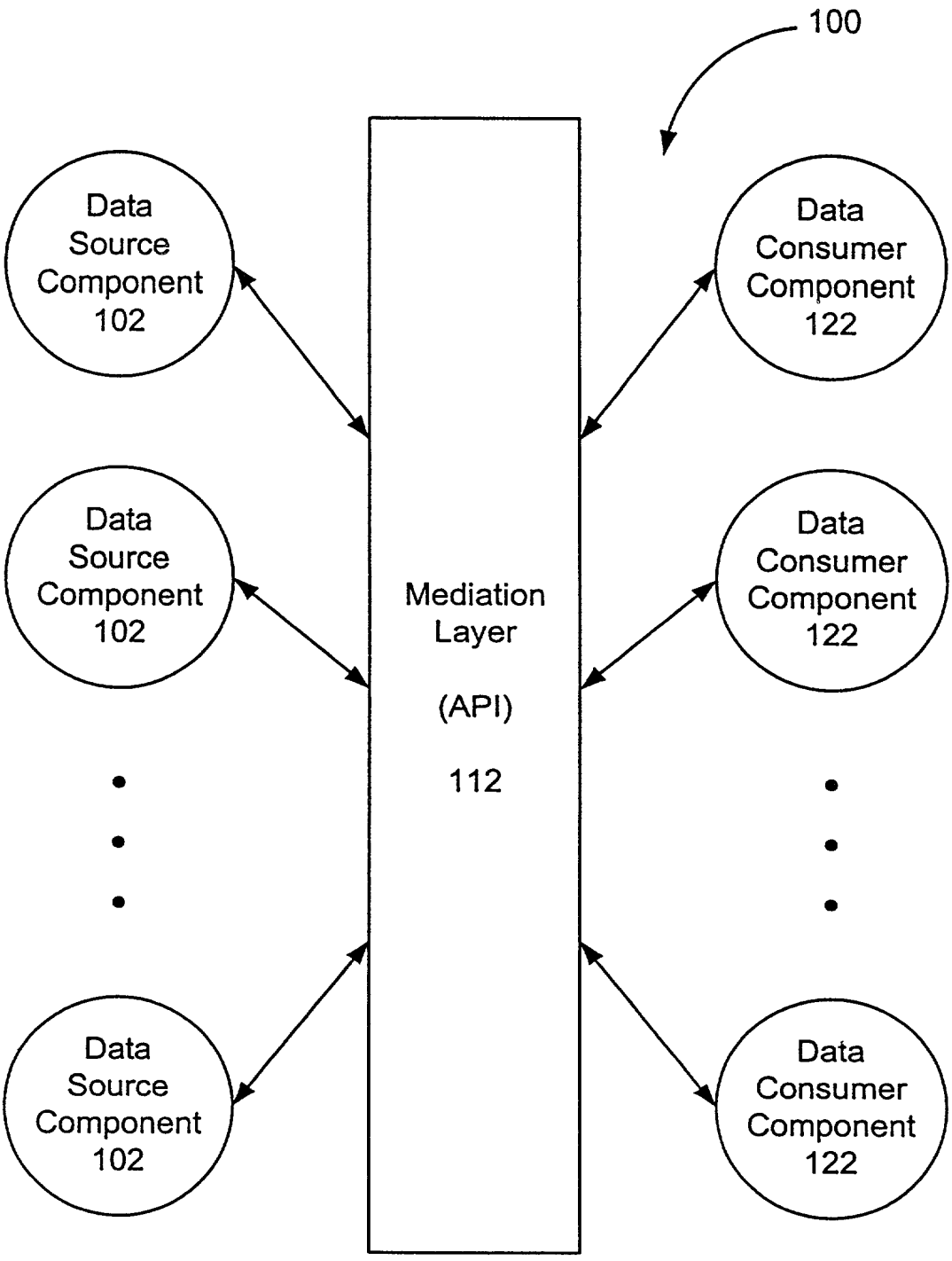

100
Data Source Component 102
Data Source Component 102
Data Source Component 102
Mediation Layer (API) 112
Data Consumer Component 122
Data Consumer Component 122
Data Consumer Component 122

FIG. 1

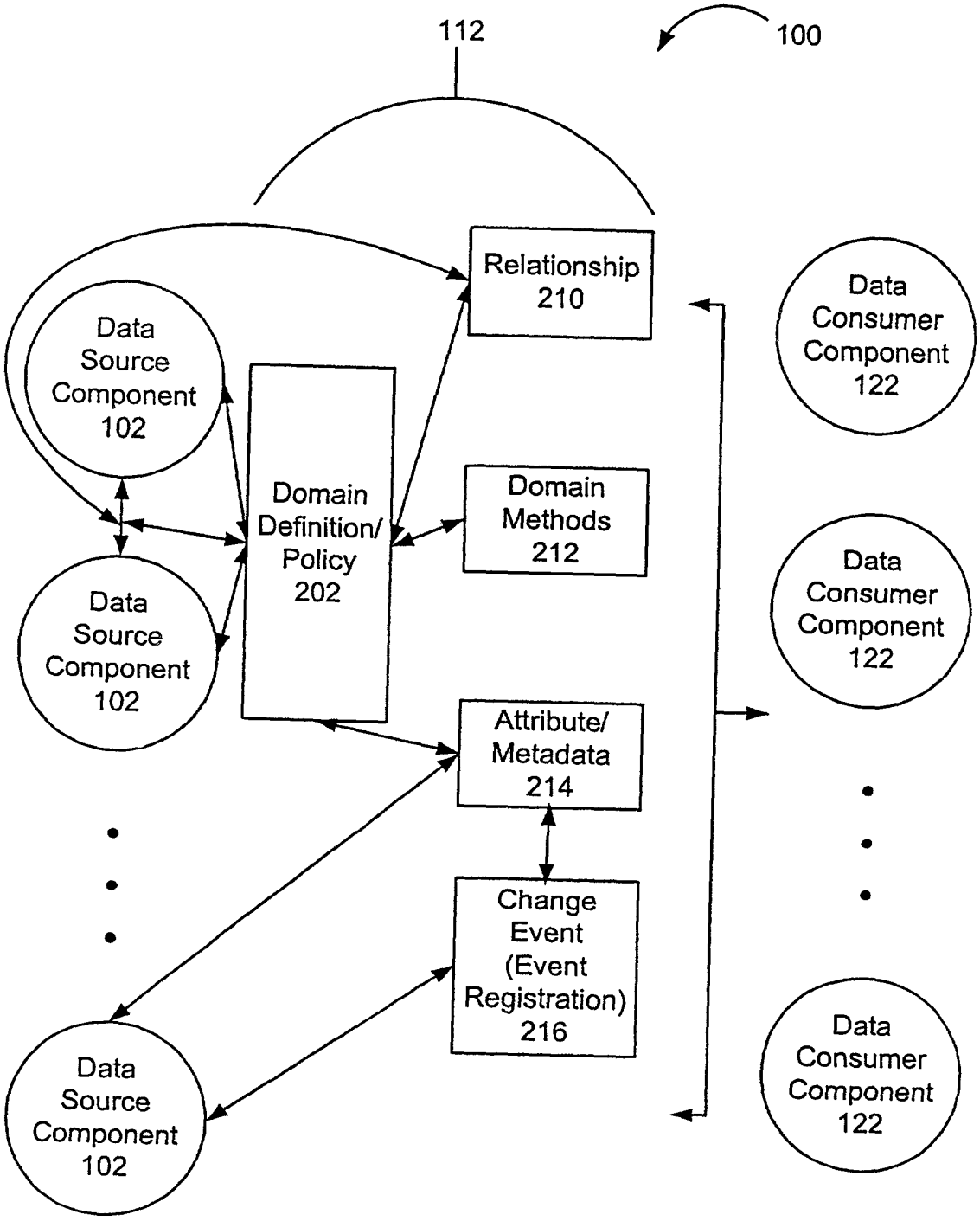
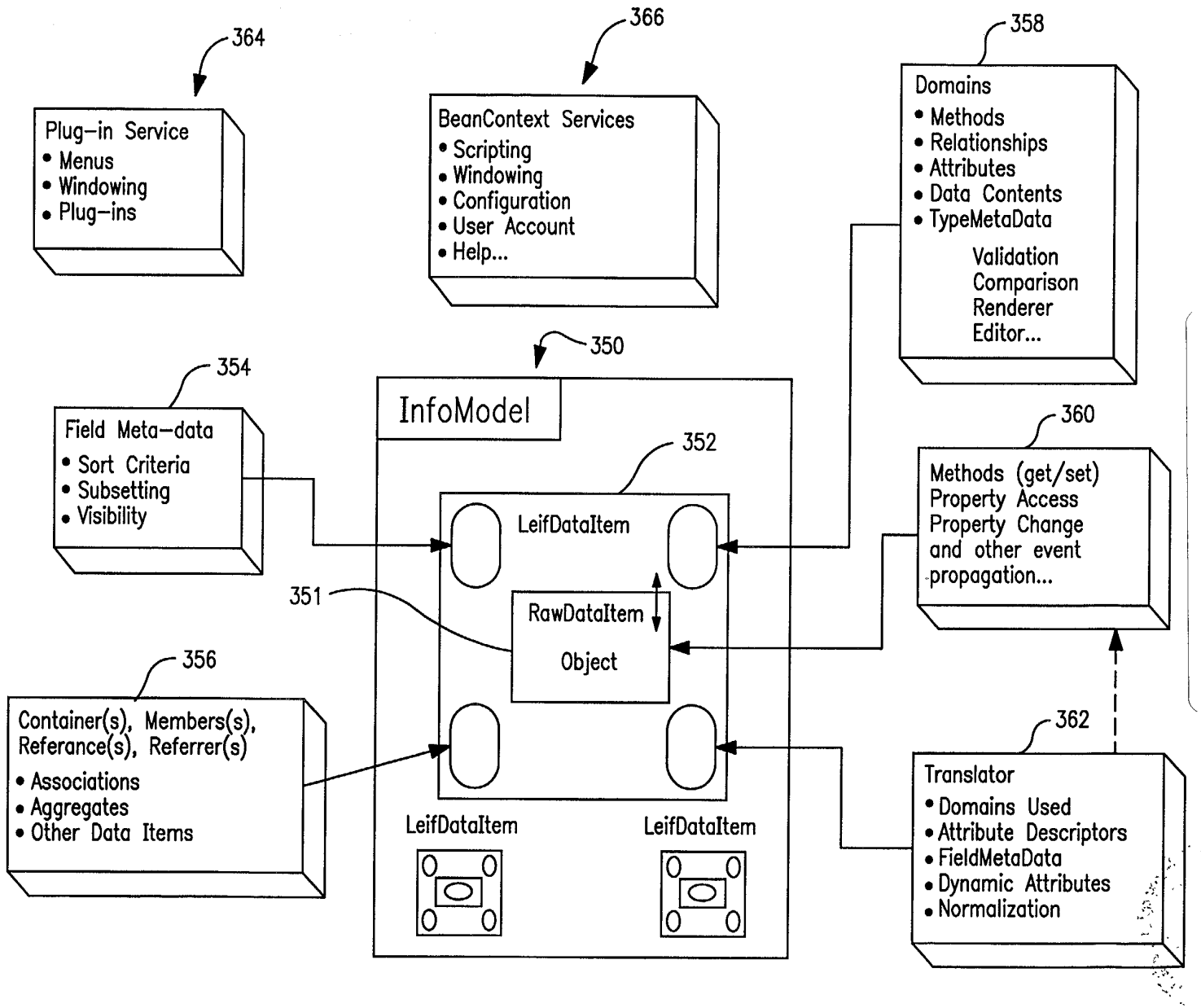

112
100
Relationship 210
Data Source Component 102
Domain Definition/ Policy 202
Domain Methods 212
Data Source Component 102
Attribute/ Metadata 214
Change Event (Event Registration) 216
Data Source Component 102
Data Consumer Component 122
Data Consumer Component 122
Data Consumer Component 122

FIG. 2

FIG. 3A


364
Plug–in Service
• Menus
• Windowing
• Plug–ins
366
BeanContext Services
• Scripting
• Windowing
• Configuration
• User Account
• Help...
358
Domains
• Methods
• Relationships
• Attributes
• Data Contents
• TypeMetaData
Validation
Comparison
Renderer
Editor...
350
InfoModel
354
Field Meta–data
• Sort Criteria
• Subsetting
• Visibility
352
LeifDataItem
351
RawDataItem
Object
360
Methods (get/set)
Property Access
Property Change
and other event
propagation...
356
Container(s), Members(s),
Referance(s), Referrer(s)
• Associations
• Aggregates
• Other Data Items
LeifDataItem
LeifDataItem
362
Translator
• Domains Used
• Attribute Descriptors
• FieldMetaData
• Dynamic Attributes
• Normalization

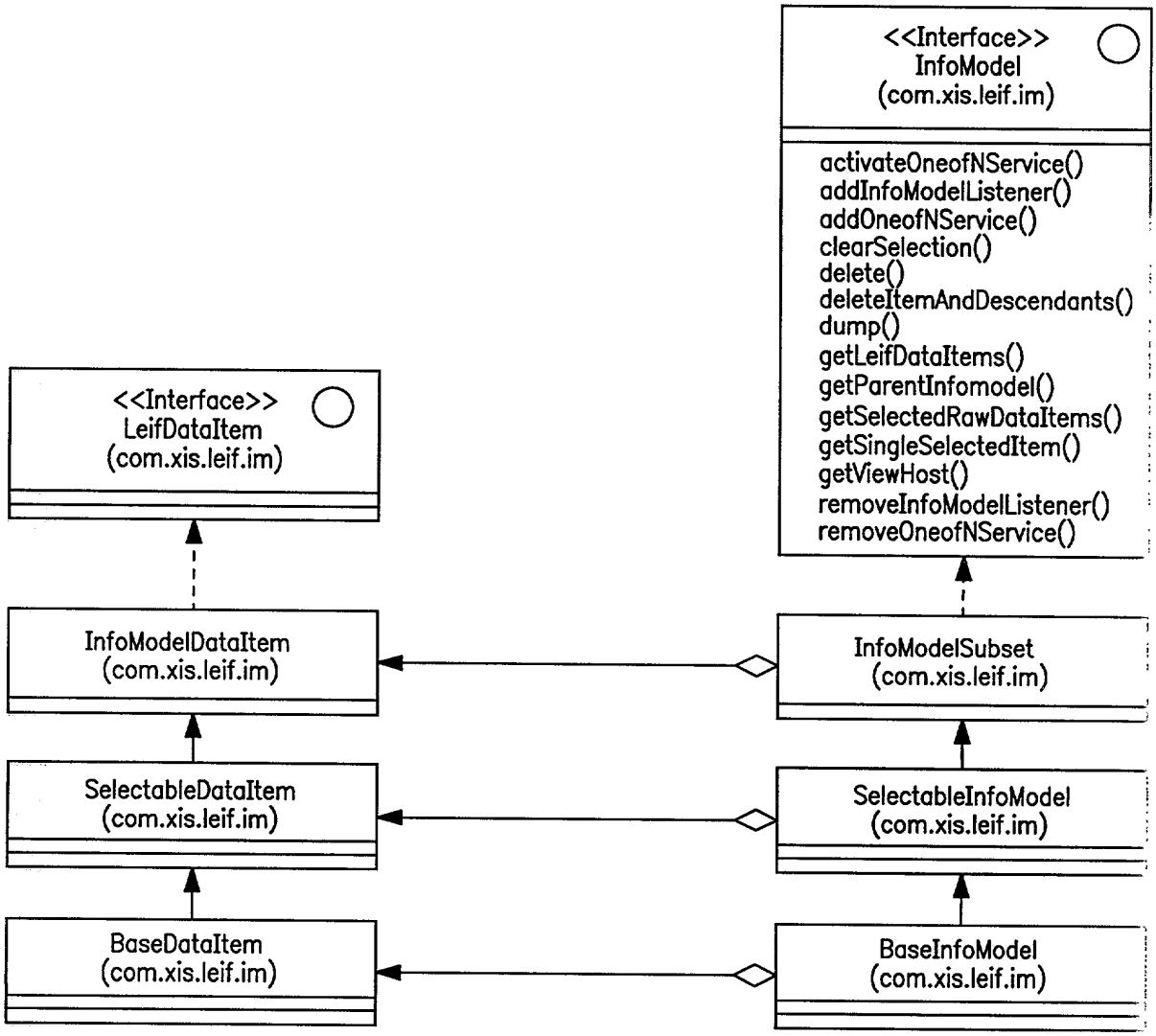

<<Interface>>
InfoModel
(com.xis.leif.im)
activateOneofNService()
addInfoModelListener()
addOneofNService()
clearSelection()
delete()
deleteItemAndDescendants()
dump()
getLeifDataItems()
getParentInfomodel()
getSelectedRawDataItems()
getSingleSelectedItem()
getViewHost()
removeInfoModelListener()
removeOneofNService()
<<Interface>>
LeifDataItem
(com.xis.leif.im)
InfoModelDataItem
(com.xis.leif.im)
InfoModelSubset
(com.xis.leif.im)
SelectableDataItem
(com.xis.leif.im)
SelectableInfoModel
(com.xis.leif.im)
BaseDataItem
(com.xis.leif.im)
BaseInfoModel
(com.xis.leif.im)

FIG. 3B

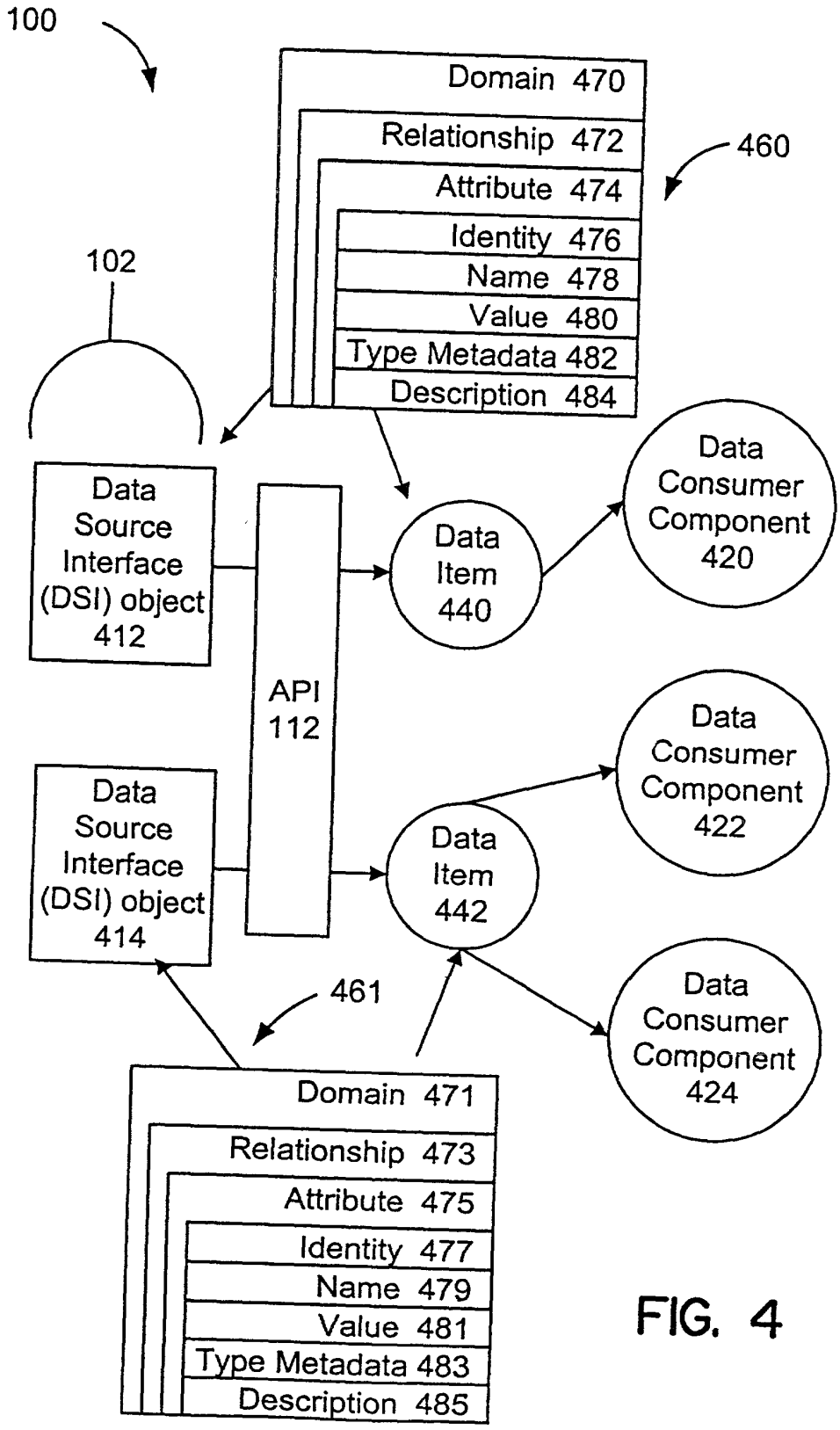
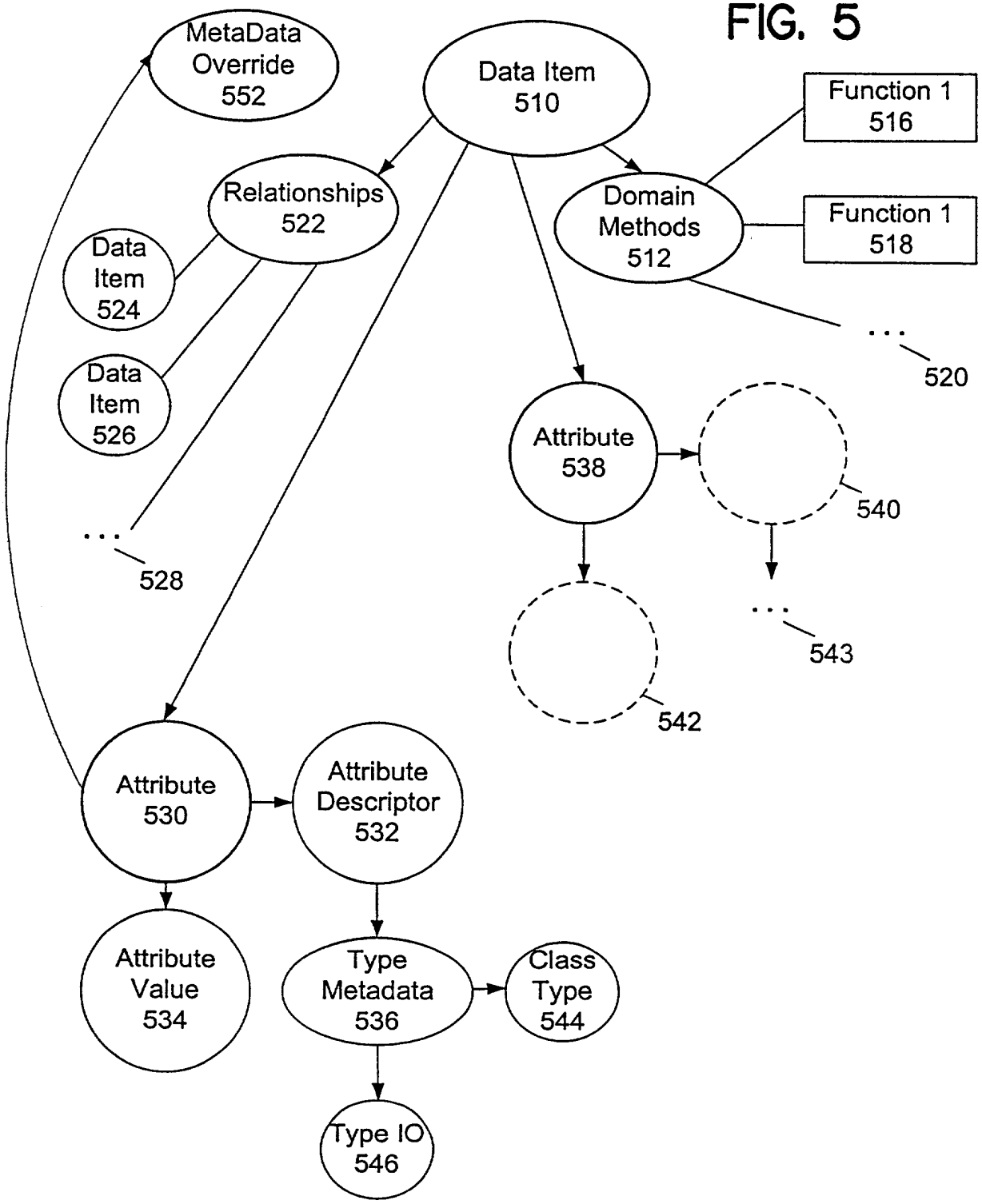

100
Domain 470
Relationship 472
Attribute 474
Identity 476
Name 478
Value 480
Type Metadata 482
Description 484
460
102
Data Source Interface (DSI) object 412
API 112
Data Item 440
Data Consumer Component 420
Data Source Interface (DSI) object 414
Data Item 442
Data Consumer Component 422
Data Consumer Component 424
461
Domain 471
Relationship 473
Attribute 475
Identity 477
Name 479
Value 481
Type Metadata 483
Description 485

FIG. 4

FIG. 5


MetaData Override 552
Data Item 510
Function 1 516
Relationships 522
Domain Methods 512
Function 1 518
Data Item 524
Data Item 526
528
520
Attribute 538
540
543
542
Attribute 530
Attribute Descriptor 532
Attribute Value 534
Type Metadata 536
Class Type 544
Type IO 546

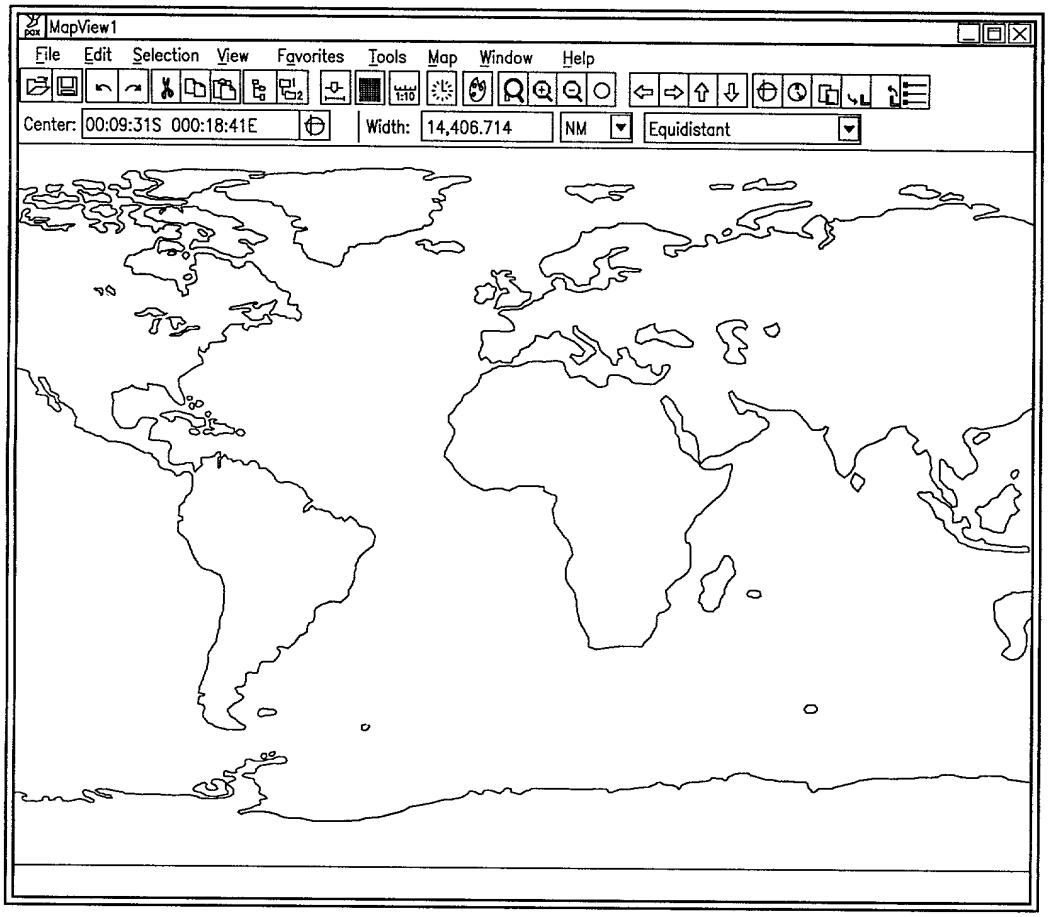

MapView1
File Edit Selection View Favorites Tools Map Window Help
Center: 00:09:31S 000:18:41E
Width: 14,406.714
NM
Equidistant

FIG. 6A

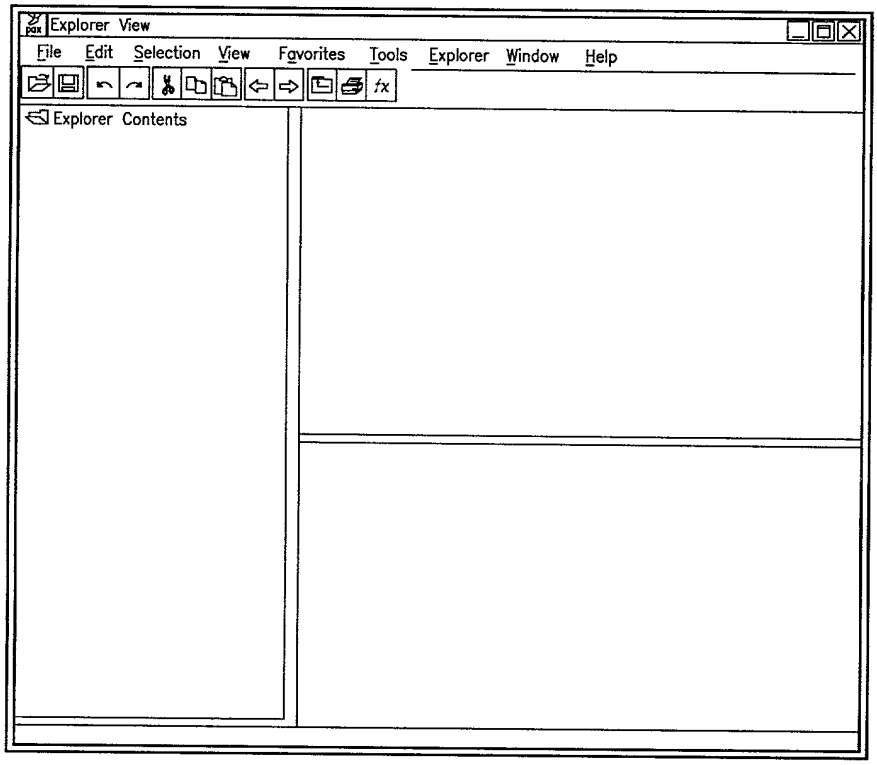

Explorer View
File Edit Selection View Favorites Tools Explorer Window Help
fx
Explorer Contents

FIG. 6B

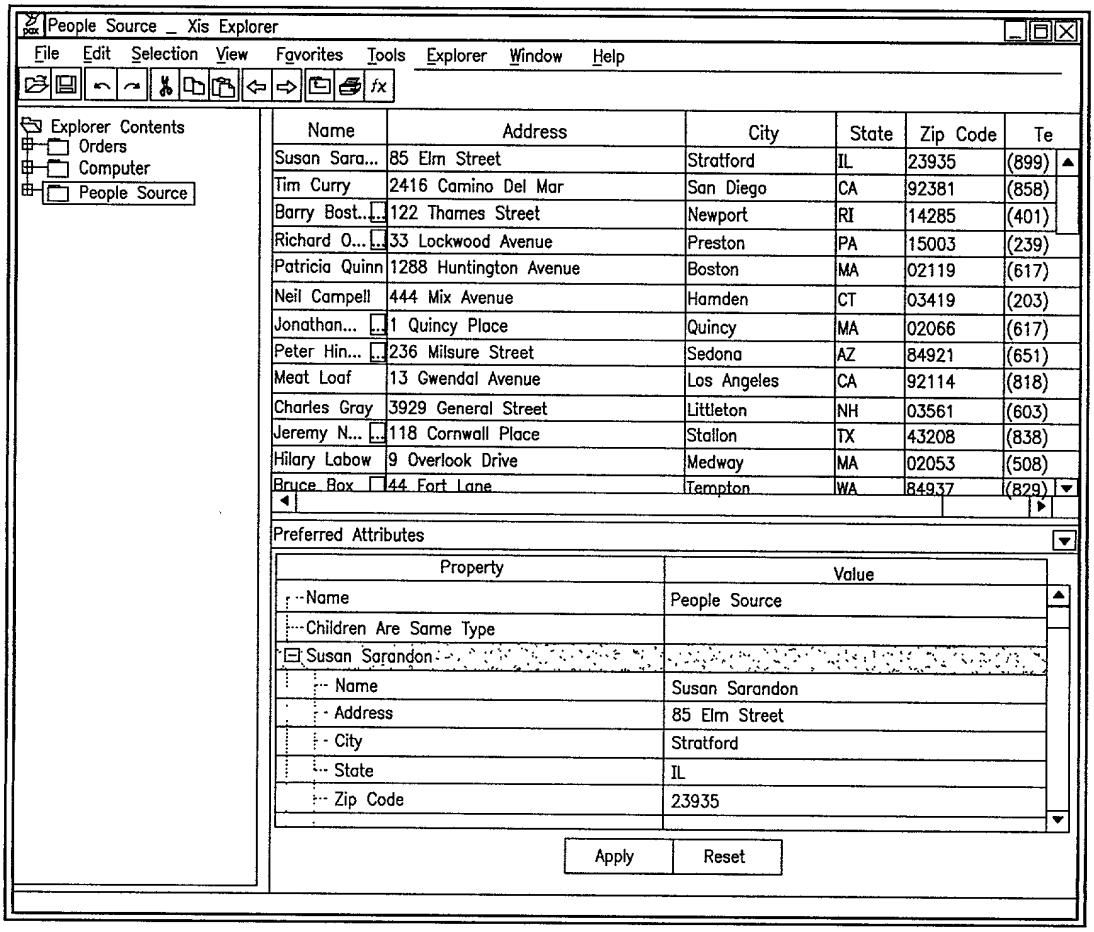

People Source _ Xis Explorer
File Edit Selection View Favorites Tools Explorer Window Help
Explorer Contents
Orders
Computer
People Source
Name | Address | City | State | Zip Code | Te
Susan Sara... | 85 Elm Street | Stratford | IL | 23935 | (899)
Tim Curry | 2416 Camino Del Mar | San Diego | CA | 92381 | (858)
Barry Bost... | 122 Thames Street | Newport | RI | 14285 | (401)
Richard O... | 33 Lockwood Avenue | Preston | PA | 15003 | (239)
Patricia Quinn | 1288 Huntington Avenue | Boston | MA | 02119 | (617)
Neil Campell | 444 Mix Avenue | Hamden | CT | 03419 | (203)
Jonathan... | 1 Quincy Place | Quincy | MA | 02066 | (617)
Peter Hin... | 236 Milsure Street | Sedona | AZ | 84921 | (651)
Meat Loaf | 13 Gwendal Avenue | Los Angeles | CA | 92114 | (818)
Charles Gray | 3929 General Street | Littleton | NH | 03561 | (603)
Jeremy N... | 118 Cornwall Place | Stallon | TX | 43208 | (838)
Hilary Labow | 9 Overlook Drive | Medway | MA | 02053 | (508)
Bruce Box | 44 Fort Lane | Tempton | WA | 84937 | (829)
Preferred Attributes
Property | Value
Name | People Source
Children Are Same Type
Susan Sarandon
Name | Susan Sarandon
Address | 85 Elm Street
City | Stratford
State | IL
Zip Code | 23935
Apply
Reset

FIG. 6C

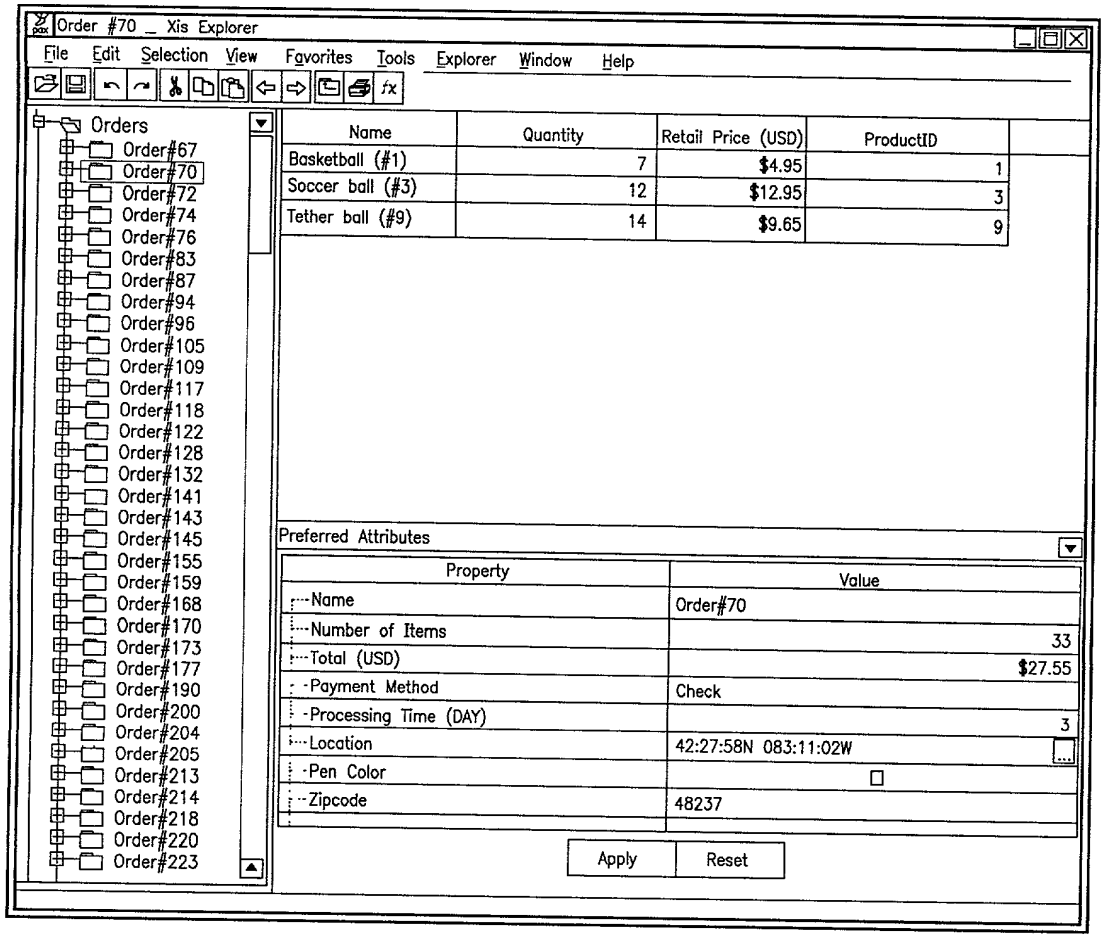

Order #70 _ Xis Explorer
File Edit Selection View Favorites Tools Explorer Window Help
Orders
Order#67
Order#70
Order#72
Order#74
Order#76
Order#83
Order#87
Order#94
Order#96
Order#105
Order#109
Order#117
Order#118
Order#122
Order#128
Order#132
Order#141
Order#143
Order#145
Order#155
Order#159
Order#168
Order#170
Order#173
Order#177
Order#190
Order#200
Order#204
Order#205
Order#213
Order#214
Order#218
Order#220
Order#223
Name | Quantity | Retail Price (USD) | ProductID
Basketball (#1) | 7 | $4.95 | 1
Soccer ball (#3) | 12 | $12.95 | 3
Tether ball (#9) | 14 | $9.65 | 9
Preferred Attributes
Property | Value
Name | Order#70
Number of Items | 33
Total (USD) | $27.55
Payment Method | Check
Processing Time (DAY) | 3
Location | 42:27:58N 083:11:02W
Pen Color
Zipcode | 48237
Apply
Reset

FIG. 6D

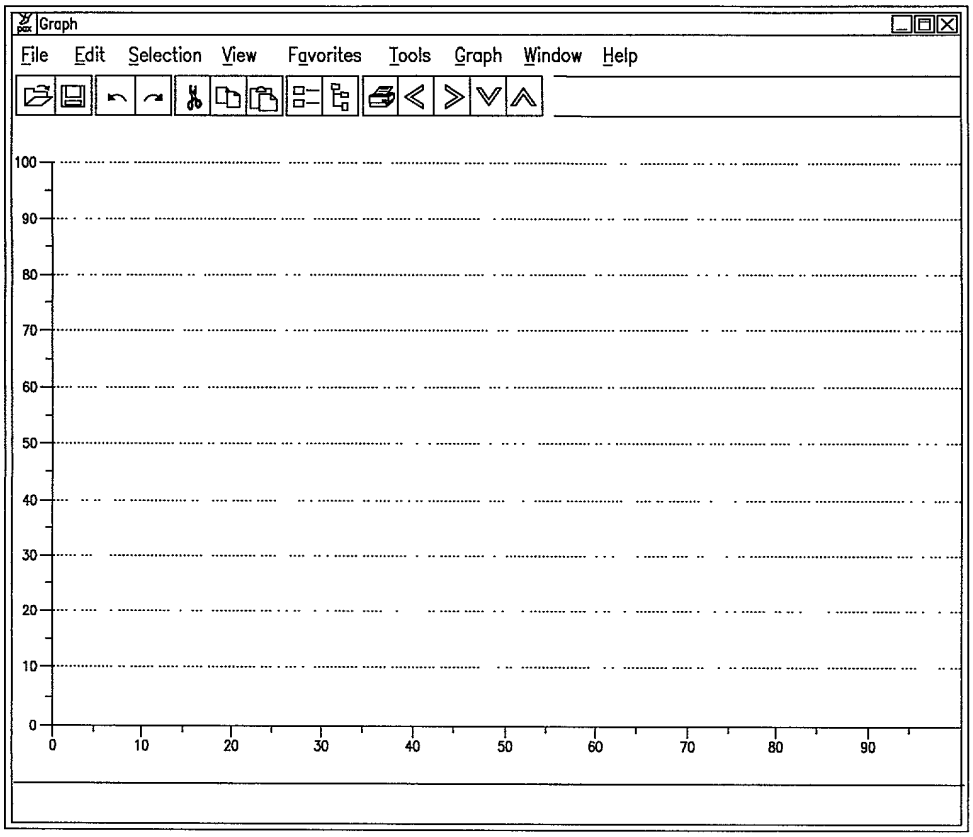

Graph
File Edit Selection View Favorites Tools Graph Window Help
100
90
80
70
60
50
40
30
20
10
0
0 10 20 30 40 50 60 70 80 90

FIG. 6E

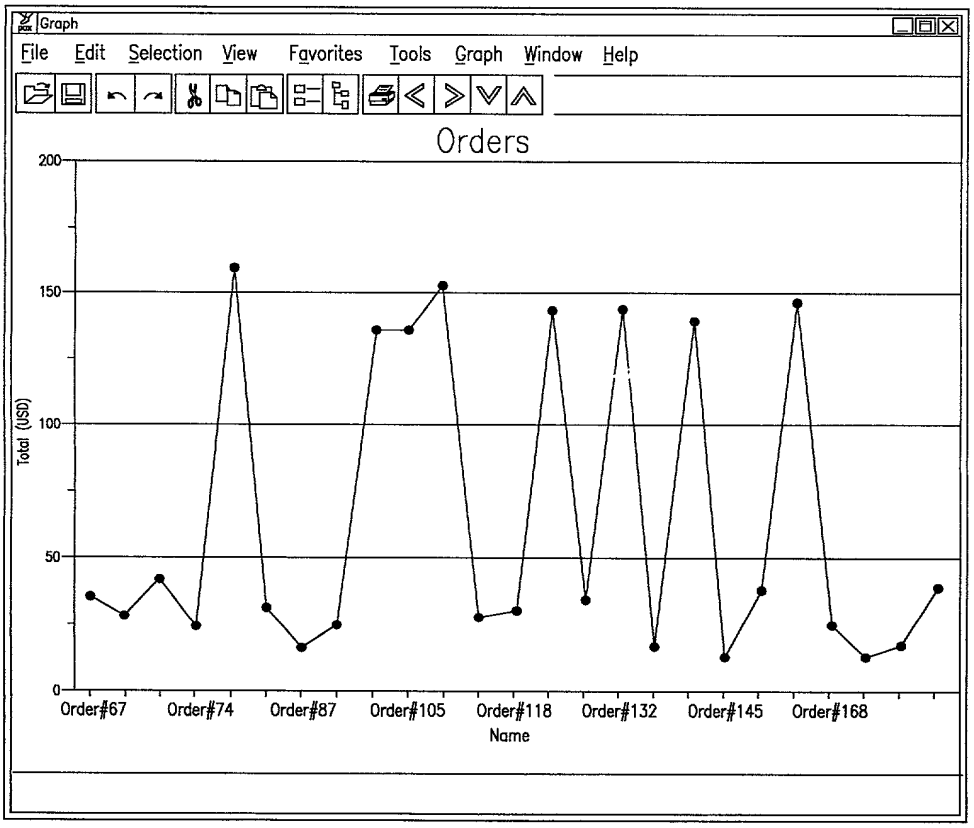

Graph
File Edit Selection View Favorites Tools Graph Window Help
Orders
200
150
100
50
0
Total (USD)
Order#67 Order#74 Order#87 Order#105 Order#118 Order#132 Order#145 Order#168
Name

FIG. 6F

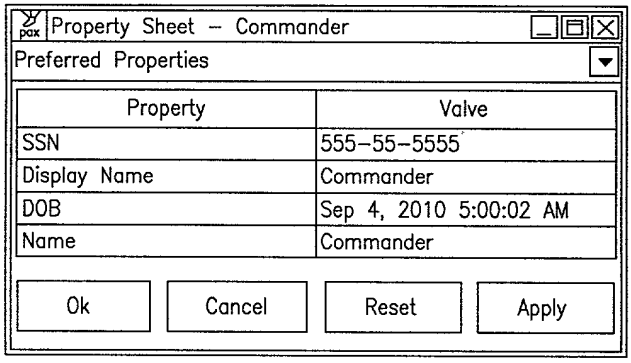

Property Sheet – Commander
Preferred Properties
Property
Valve
SSN
555–55–5555
Display Name
Commander
DOB
Sep 4, 2010 5:00:02 AM
Name
Commander
Ok
Cancel
Reset
Apply

FIG. 7

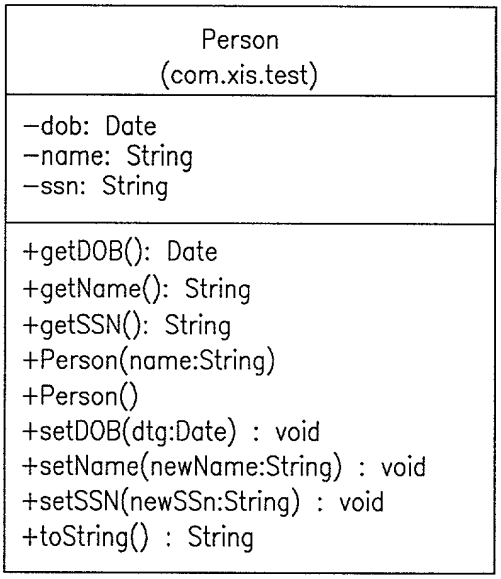

Person
(com.xis.test)
–dob: Date
–name: String
–ssn: String
+getDOB(): Date
+getName(): String
+getSSN(): String
+Person(name:String)
+Person()
+setDOB(dtg:Date) : void
+setName(newName:String) : void
+setSSN(newSSn:String) : void
+toString() : String

FIG. 8

getAttributes()	References	Referrers	Members
com.xis.domains.display.DisplayDomain.displayName com.xis.test.Person.DOB com.xis.test.Person.SSN com.xis.test.Person.name	None	None	None

FIG. 9

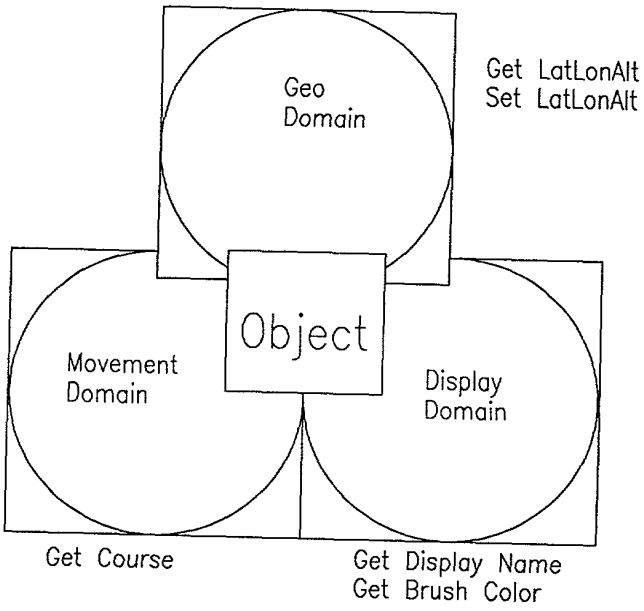


FIG. 10

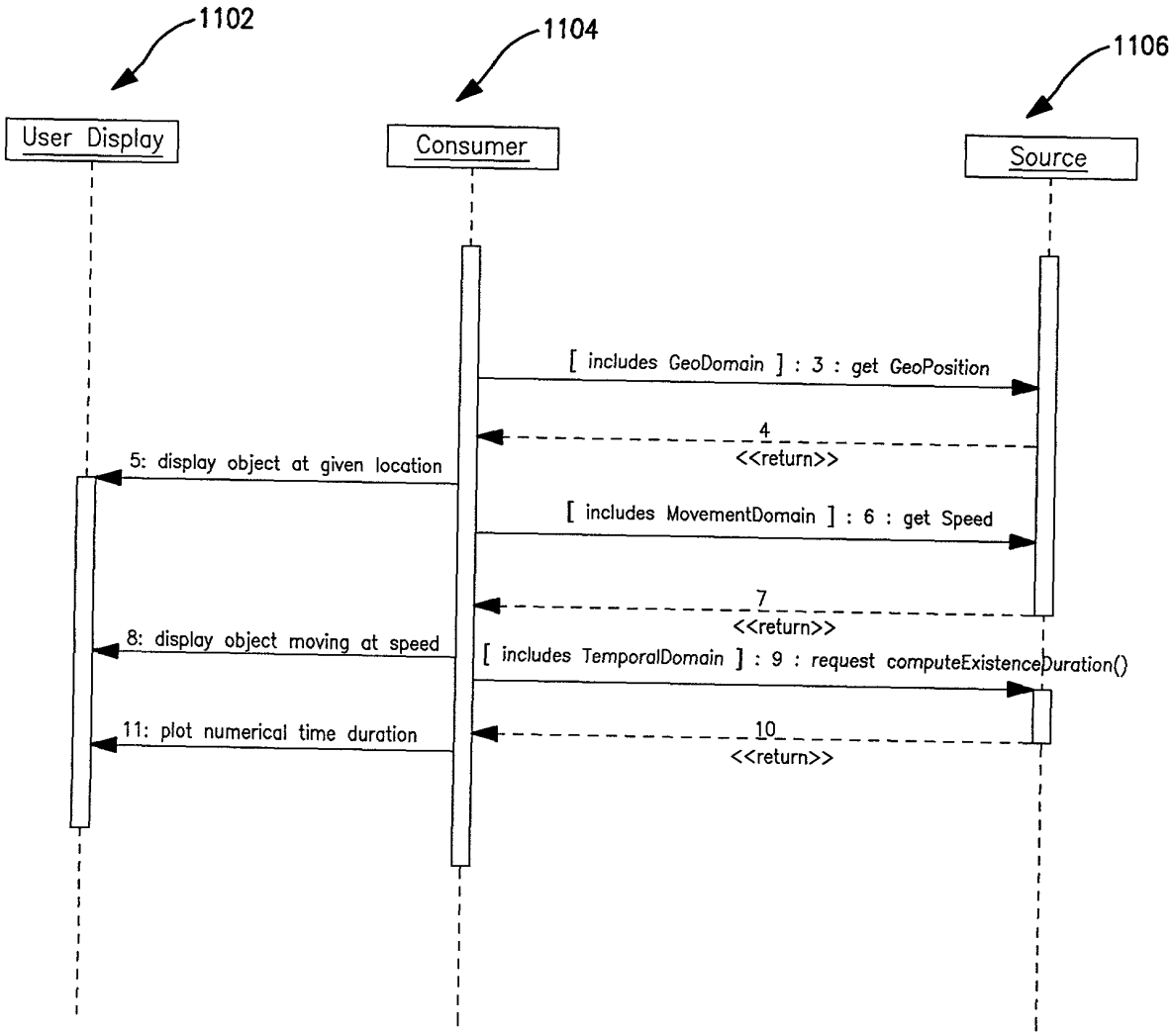

1102
1104
1106
User Display
Consumer
Source
[includes GeoDomain] : 3 : get GeoPosition
4
<<return>>
5: display object at given location
[includes MovementDomain] : 6 : get Speed
7
<<return>>
8: display object moving at speed
[includes TemporalDomain] : 9 : request computeExistenceDuration()
11: plot numerical time duration
10
<<return>>

FIG. 11

Package com.xis.types

This package contains classes that provide several standard TypeMetaData classes for describing types and their constraints, and for rendering and editing values of those types.

See:
Description

Interface Summary	
DataTest	The DataTest interface specifies methods for Object validation.
HTMLTypeIO	This interface defines the IO for HTML.
Summary Function	The SummaryFunction interface defines generic summary functionality based upon provided input data values.
SwingTypeIO	The SwingTypeIO interface allows for the use of both swing editors, allowing swing components to edit an object, and swing renderers, which know how to render these objects in a swing environment.
TextTypeIO	The TextTypeIO interface provides a means of formatting objects in a textual fashion, as well as parsing text from which an object is created.
TypedValue	The TypeValue interface is used to hold an object that carries its own TypeMetaData with it.
TypeEditor	The TypeEditor interface defines methods for editing attributes provided by the types implemented within this package.
TypeIO	The TypeIO interface provides a common base from which other TypeIOs can extend, such as HTMLTypeIO, SwingTypeIO,etc.
TypeMetaData	The TypeMetaData interface defines generic type accessors for object comparing, editing, formatting, rendering, and validation.
TypeMetaDataFactory	The TypeMetaDataFactory interface defines a class that can create TypeMetaData for a given Class Type.
TypeRenderer	The TypeRenderer interface defines methods for rendering attributes provided by the types implemented within this package.
ValidTestProxy	The ValidTestProxy interface.
WMLTypeEditor	The WMLTypeEditor interface defines methods for rendering attributes provided by the types implemented within this package.
WMLTypeIO	This interface defines the IO for WML format.
WMLTypeRenderer	The WMLTypeRenderer interface defines methods for rendering attributes provided by the types implemented within this package.
XMLTypeIO	The XMLTypeIO interface provides a means of formatting objects in a XML textual fashion, as well as parsing XML text for creating an object.

FIG. 12A

Class Summary	
AbstractDataTest	The AbstractDataTest class provides a default implementation of *test (Object)*
AbstractTypeMetaData	AbstractTypeMetaData provides a partial implementation of TypeMetaData to relieve the XIS developer from explicitly implementing irrelevant methods.
AreaOfUncertaintyTypeMetaData	AreaOfUncertaintyTypeMetaData is a type object that supports *AreaOfUncertainty objects.*
ArrayListTypeMetaData	ArrayListTypeMetaData is a type object that supports *java.util.ArrayList objects.*
ArrayTypeMetaData	ArrayTypeMetaData is a type object that supports *java.lang.reflect.Array* objects.
BeanTypeMetaData	BeanTypeMetaData is a type object that supports *java.beans* objects.
BooleanTypeMetaData	The BooleanTypeMetaData is a type object that supports *Boolean* objects.
BooleanTypeMetaDataFactory	A BooleanTypeMetaDataFactory can create a BooleanTypeMetaData for given booleans.
CachedTypeMetaData	CachedTypeMetaData is a type object that simply delegates all TypeMetaData calls to another TypeMetaData.
ClassificationTypeMetaData	ClassificationTypeMetaData is a type object that supports *Classification* objects.
CollectionsTypeMetaData	CollectionsTypeMetaData is a type object that supports *Collection* objects.
ColorTypeMetaData	The ColorTypeMetaData class implements TypeMetaData for Color objects.
ColorTypeMetaDataFactory	A ColorTypeMetaDataFactory can create a ColorTypeMetaData for a given Color object.
ConversionNumericTypeMetaData	A generic NumericTypeMetaData for converting from one unit to another.
CurrencyTypeMetaData	CurrencyTypeMetaData is a type object that supports *Number* objects that represent Currency values.
DateTimeTypeMetaData	DataTimeTypeMetaData is a type object that supports *Date* objects.
DataTimeTypeMetaDataFactory	A DateTimeTypeMetaDataFactory can create a DataTimeTypeMetaData for given Date objects.
DiscreteRangeStringTypeMetaData	DiscreteRangeStringTypeMetaData is a type object that supports *String* objects with discrete ranges.
DisplayLabelTypeMetaData	DisplayLabelTypeMetaData is a type object that supports DisplayLabel objects
DTGTypeMetaData	DTGTypeMetaData is a type object that supports *Date* objects.
EnumerationType	Class to implement an enumeration in Java.
EnumerationTypeMetaData	The EnumerationTypeMetaData class is used to represent integer constants as strings to the user.
FontTypeMetaData	FontTypeMetaData is a type object that supports *Font* objects.
HashMapTypeMetaData	HashMapTypeMetaData is a type object that supports *java.util.HashMap* objects.

FIG. 12B

HashSetTypeMetaData	HashSetTypeMetaData is a type object that supports *java.util.HashSet* objects.
IconShapeTypeMetaData	IconShapeTypeMetaData is a type object that supports *IconShape* objects.
IconTypeMetaData	IconTypeMetaData is a type object that supports *Icon* objects.
LinkedListTypeMetaData	LinkedListTypeMetaData is a type object that supports *java.util.LinkedList* objects.
ListTypeMetaData	ListTypeMetaData is a type object that supports *java.util.List* objects.
MouseMapProxy	The MouseMapProxy class allows TypeEditors access to the map without requiring them to having any compile time knowledge of the map's existence.
NumberComparator	An implementation of the Comparator interface that compares two objects that extend Number, or that both implement Comparable, with the class of one assignable from the other.
NumericTypeMetaData	NumericTypeMetaData is a type object that supports *Number* objects.
NumericTypeMetaDataFactory	A NumericTypeMetaDataFactory can create NumericTypeMetaData for a given class type or method return type.
ObjectTypeMetaData	ObjectTypeMetaData is a type that supports a simple *Object* and is provided to quickly add arbitrary attribute types to a Data Source Interface without writing a more specific type handler.
PercentTypeMetaData	PercentTypeMetaData is a type object that supports *Number* objects that represent Percent(%) values.
ProbabilityTypeMetaData	ProbabilityTypeMetaData is a type object that supports *Number* objects that represent Probability values.
RenamedTypeMetaData	RenamedTypeMetaData simply delegates all TypeMetaData calls to another TypeMetaData except for the *getName()*, which is overridden with the given value.
ResizedTextTypeMetaData	ResizedTextTypeMetaData simply delegates all TypeMetaData calls to another TypeMetaData except for the *getPixelWidth()*, which is overridden with the given value.
Resources	The Resources class is automatically generated and must be public, but it is intended to be used only by Java's internationalization support classes.
SmartDurationTypeMetaData	A SmartDurationTypeMetaData for converting from one time unit to another based on the magnitude of the duration value.
StringTypeMetaData	StringTypeMetaData is a type object that supports *String* objects.
StringTypeMetaDataFactory	A StringTypeMetaDataFactory can create a StringTypeMetaData for a given String object.
TextTypeMetaData	TextTypeMetaData is a type object that supports *Text* objects.
TypedValueTypeMetaData	TypedValueTypeMetaData is a type object that supports *TypedValue* objects.
TypedEditorBeanContextChildSupport	The TypeEditorBeanContextChildSupport class handles most of the responsibilities of a TypeEditor and a BeanContextChild.
TypedEditorSupport	The TypeEditorSupport class handles most of the responsiblities of a TypeEditor.
TypeIOPluggableService	The TypeIOPluggableService class is responsible for loading TypeIOs.
TypeIORegistry	The TypeIORegistry class is a registry for different implementations of TypeIO classes.

FIG. 12C

TypeIOSupport	The TypeIOSupport provides support TypeMetaData's get TypeIO methods.
TypeMetaDataDelegator	TypeMetaDataDelegator is a type object that simply delegates all TypeMetaData calls to another TypeMetaData.
TypeMetaDataFactoryPluggableService	The TypeMetaDataFactoryPluggableService class is responsible for loading TypeMetaDataFactories.
TypeMetaDataFactoryStore	This Class stores TypeMetaDataFactories.
TypePreferences	This class is used by TypeMetaData instances to pass information about preferred TypeMetaData objects.
Types	The Types class is a holder class for the RESOURCES global variable for resource properties of the com.xis.types package.
URLTypeMetaData	URLTypeMetaData is a type object that supports *URL* objects.

Exception Summary	
NoSuchEnumerationException	Class to implement an enumeration exception in Java.
ParseFailedException	A ParseFailedException is thrown (typically by TypeIO objects) when it is not possible to parse a given String as described.
TestFailedException	The TestFailedException is thrown from the "test()" method of a DataTest subclass when the value fails the test.

Package com.xis.types Description

This package contains classes that provide several standard TypeMetaData classes for describing types and their constraints, and for rendering and editing values of those types.

FIG. 12D

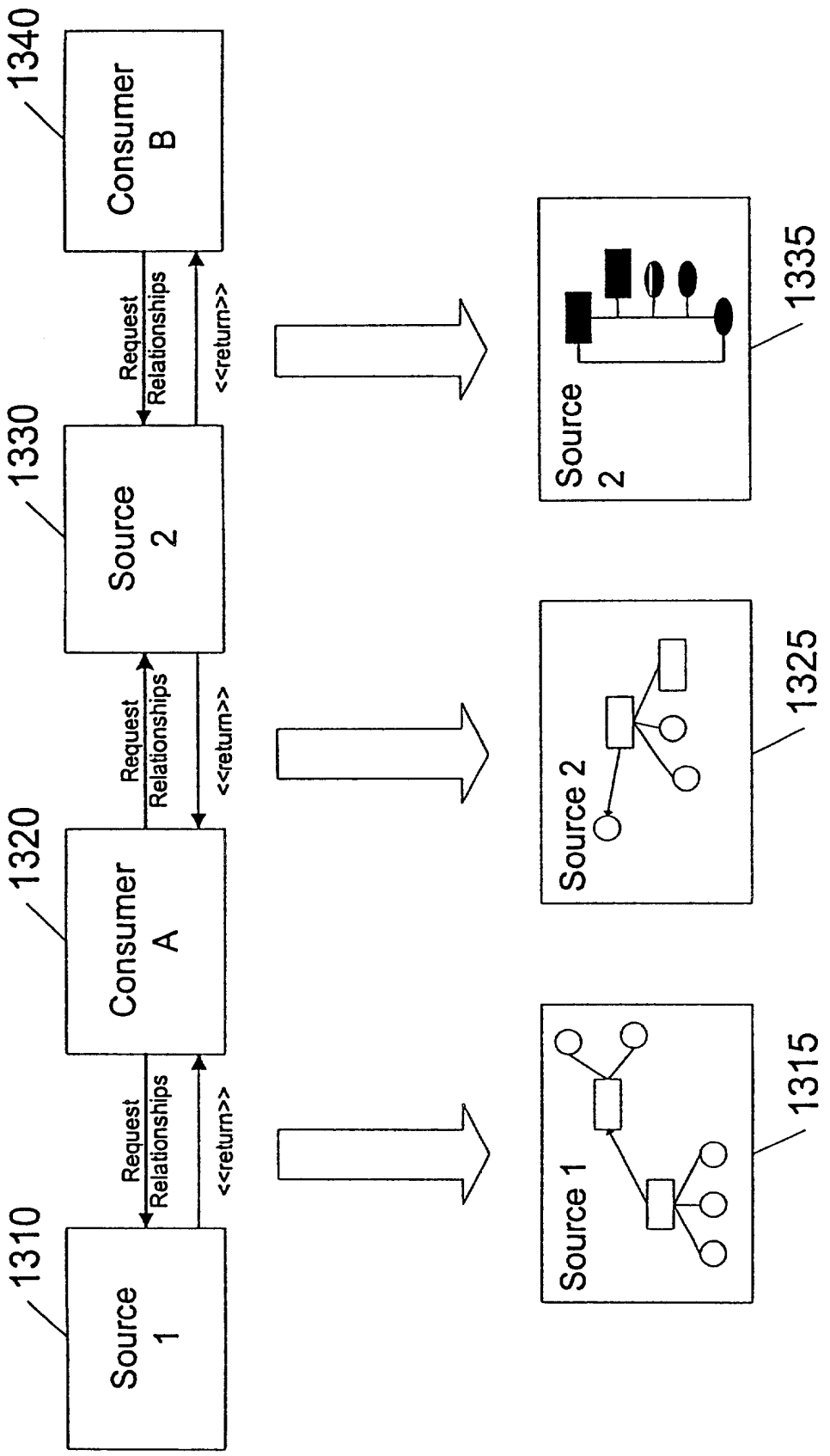

1340
Consumer B
1330
Source 2
1320
Consumer A
1310
Source 1
Request Relationships
<<return>>
Request Relationships
<<return>>
Request Relationships
<<return>>
Source 2
1335
Source 2
1325
Source 1
1315

FIG. 13

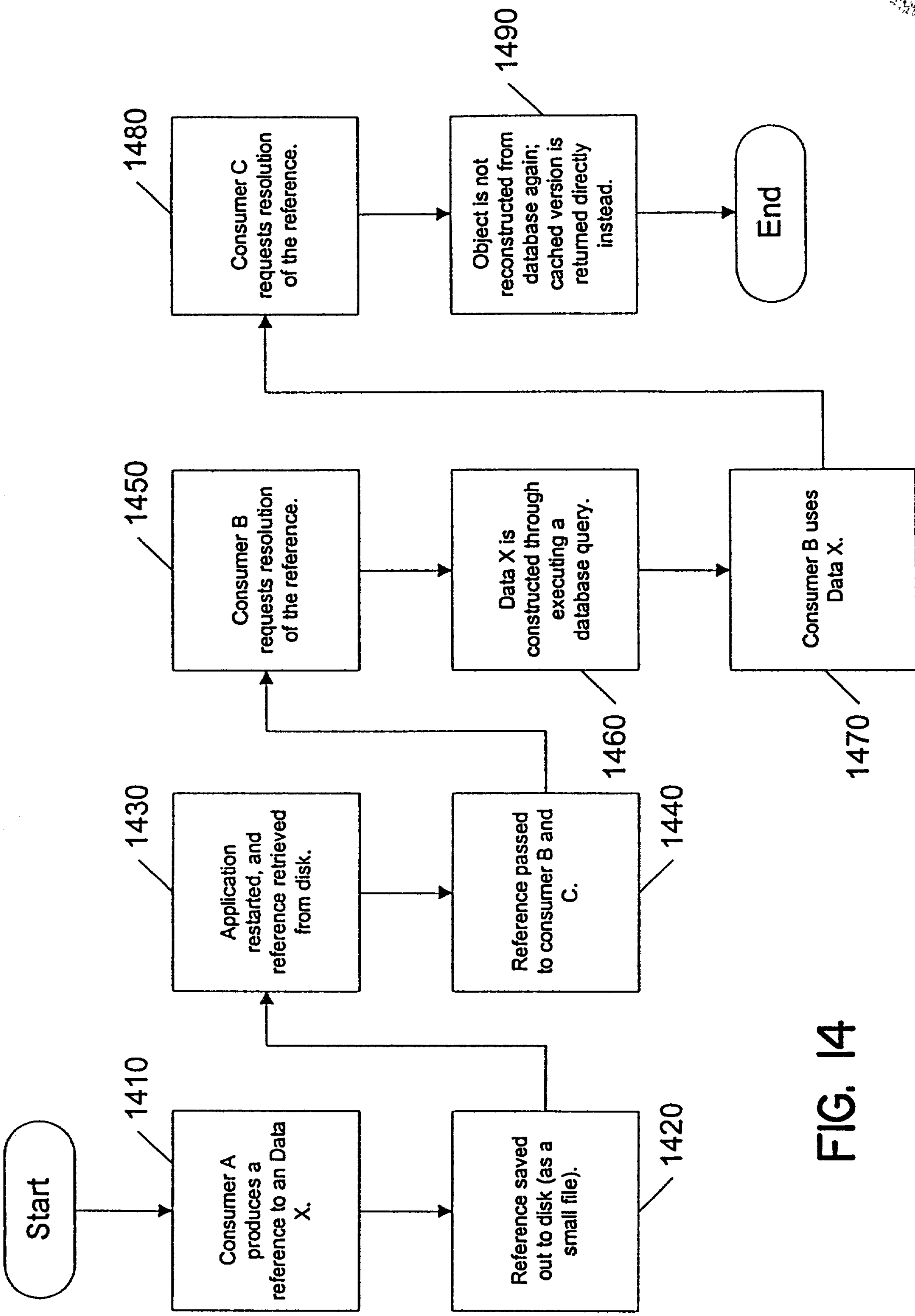

Start
Consumer A produces a reference to an Data X.
1410
Reference saved out to disk (as a small file).
1420
Application restarted, and reference retrieved from disk.
1430
Reference passed to consumer B and C.
1440
Consumer B requests resolution of the reference.
1450
Data X is constructed through executing a database query.
1460
Consumer B uses Data X.
1470
Consumer C requests resolution of the reference.
1480
Object is not reconstructed from database again; cached version is returned directly instead.
1490
End

FIG. 14

HelloWorld.java

```
/* XIS Tutorial standalone sequence example 1 data class. */

public class HelloWorld {
   private float value = 1.5f;

   public String toString() {
      return "Hello World!";
   }

   public int getID() {
      return 5;
   }

   public float getValue() {
      return value;
   }

   // uncomment this to make "value" editable
   /*
   public void setValue(float value) {
      this.value = value;
   }
   */

}
```

FIG. 15

TestHarness.java

```
/* XIS Tutorial standalone sequence example 1 XIS interfacing. */
import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import com.xis.propertysheet.PropertySheetInfoBean;
import com.xis.ui.UIBeanEvent;
import com.xis.ui.UIBeanAdapter;
import com.xis.leif.im.BaseInfoModel;

public class TestHarness {

   public static void main(String[] args) {

      // the plugin manager is only required for more complex applications
      // involving multiple components integrated at runtime
      BaseInfoModel.setStartingPlugInManager(false);

      // a property sheet infobean to display HelloWorld's attributes
      PropertySheetInfoBean properties = new PropertySheetInfoBean();
      properties.addRawDataItem(new HelloWorld());

      // add listener for 'OK,' 'cancel,' or close, which generate 'close' events
      properties.addUIBeanListener(
         new UIBeanAdapter() {
            public void closed(UIBeanEvent event) {
               System.exit(0);
            }
         }
      );

      // a top-level frame to hold our property sheet infobean
      JFrame frame = new JFrame("HelloWorld Properties");
      // add a listener for window closing
      frame.addWindowListener(
         new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
               System.exit(0);
            }
         }
      );
```

1602

1604

1606

1608

1610A

FIG. 16A

Continuation of TestHarness.java

```
        // stick the bean in the frame and display it
        frame.getContentPane().add(properties);
        frame.pack();
        frame.setVisible(true);

    }
}
```

1610B

FIG. 16B

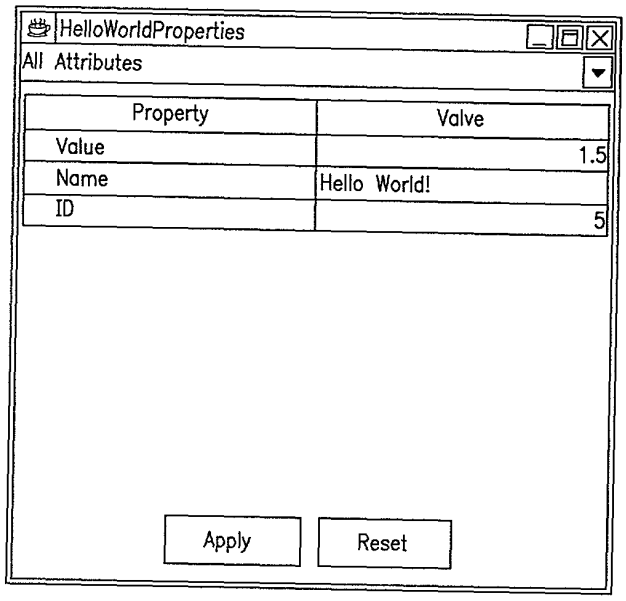

HelloWorldProperties
All Attributes
Property
Valve
Value
1.5
Name
Hello World!
ID
5
Apply
Reset

FIG. 17

HelloWorld.java

```
/* XIS Tutorial standalone sequence example 2 data class. */

/*{*/
import java.awt.Color;
import java.beans.PropertyChangeSupport;
import java.beans.PropertyChangeListener;
/*}*/

public class HelloWorld {

/*{*/
   private int value = 1;
   private Color myColor = Color.green;


   // this member class helps distribute property change events within XIS
   private PropertyChangeSupport propertyChangeSupport =
                              new PropertyChangeSupport(this);

   // two aux methods to let other XIS objects pay attention to this one
   public void addPropertyChangeListener(PropertyChangeListener l) {
      propertyChangeSupport.addPropertyChangeListener(l);
   }
   public void removePropertyChangeListener(PropertyChangeListener l) {
      propertyChangeSupport.removePropertyChangeListener(l);
   }


   public String toString() {
      return "A HelloWorld Object";
   }

   public String getGreeting() {
      return "Hello World!";
   }
/*}*/

public int getID() {
      return 5;
   }
```

1806

1802

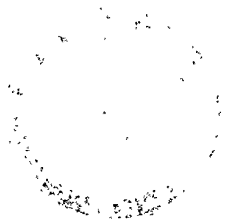
FIG. 18A

Continuation of of HelloWorld.java

```
    public /*{*/ int /*}*/ getValue() {
       return value;
    }

  /*{*/
    public void setValue(int value) {
       // only update and fire property change if this is really a change
       if (this.value != value) {
          int oldValue = this.value;
          this.value = value;
          // fire property change event to notify other XIS objects
          propertyChangeSupport.firePropertyChange("value", oldValue, value);
       }
    }

    public Color getMyColor() {
       return myColor;
    }

    public void setMyColor(Color myColor) {
       // only update and fire property change if this is really a change
       if (this.myColor != myColor) {
          Color oldMyColor = this.myColor;
          this.myColor = myColor;
          // fire property change event to notify other XIS objects
          propertyChangeSupport.firePropertyChange("myColor",
                                    oldMyColor, myColor);
       }
    }
  /*}*/

}
```

1808 1804 1810 1804

FIG. 18B



TestHarness.java

```
/* XIS Tutorial standalone sequence step 2 XIS interfacing. */

import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import com.xis.propertysheet.PropertySheetInfoBean;
import com.xis.ui.UIBeanEvent;
import com.xis.ui.UIBeanAdapter;
import com.xis.leif.im.BaseInfoModel;
/*{*/
import jclass.chart.JCChart;
import com.xis.plot.PlotInfoBean;
import com.xis.plot.chartviews.LeifChartView;
/*}*/

public class TestHarness {

   public static void main(String[] args) {

      // the plugin manager is only required for more complex applications
      // involving multiple components integrated at runtime
      BaseInfoModel.setStartingPlugInManager(false);

      HelloWorld hello = new HelloWorld();

      // a property sheet infobean to display HelloWorld's attributes
      PropertySheetInfoBean properties = new PropertySheetInfoBean();
      properties.addRawDataItem(hello);

      // add a listener for 'OK' or 'cancel', which generate 'close' events
      properties.addUIBeanListener(
         new UIBeanAdapter() {
            public void closed(UIBeanEvent event) {
               System.exit(0);
            }
         }
      );
```

1902

FIG. 19A

Continuation of TestHarness.java

```
/*{*/
    // a top-level frame to hold our property sheet infobean
    JFrame propertySheetFrame = new JFrame("HelloWorld Properties");
    // add a listener for window closing
    propertySheetFrame.addWindowListener(
       new WindowAdapter() {
          public void windowClosing(WindowEvent e) {
             System.exit(0);
          }
       }
    );

    // stick the property Sheet bean in the frame and display it
    propertySheetFrame.getContentPane().add(properties);
    propertySheetFrame.pack();
    propertySheetFrame.setVisible(true);

    // now we create a plot infobean to plot HelloWorld's numeric attribute
    PlotInfoBean plot = new PlotInfoBean();
    plot.addRawDataItems(new Object[] { hello });
    plot.setChartType(JCChart.BAR);
    // the alternatives are SCATTER_PLOT, PLOT, AREA, PIE, CANDLE,
    // and STACKING_BAR, though not all will make sense in this example

    // a top-level frame as before to hold our property sheet infobean
    JFrame plotFrame = new JFrame("HelloWorld Plot");

    // stick the plot bean in and put it up
    plotFrame.getContentPane().add(plot);
    plotFrame.pack();
    plotFrame.setVisible(true);
/*}*/

  }

}
```

1904

1906

FIG. 19B

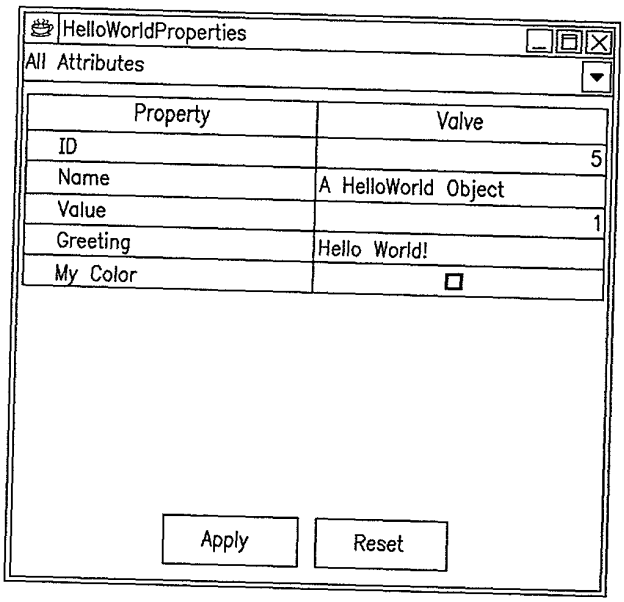

HelloWorldProperties
All Attributes
Property
Valve
ID
5
Name
A HelloWorld Object
Value
1
Greeting
Hello World!
My Color
Apply
Reset

FIG. 20

HelloWorld.java

```
/* XIS Tutorial standalone sequence step 3 data class. */
import java.awt.Color;
// (property change support moved to HelloWorldTranslator)

public class HelloWorld {
   private int value = 1;
   private Color myColor = Color.green;

   public String toString() {
      return "A HelloWorld Object";
   }

   public String getGreeting() {
      return "Hello World!";
   }

   public int getID() {
      return 5;
   }

   public int getValue() {
      return value;
   }

/*{*/
   public void setValue(int value) {
      // all the worrying about change events is moved to the translator,
      // so we just need to do the bare change operation (unless nonXIS
      // components need to listen to PropertyChanges)
      this.value = value;
   }
/*}*/

   public Color getMyColor() {
      return myColor;
   }

/*{*/
   public void setMyColor(Color myColor) {
      // just need to set the value (see setValue())
      this.myColor = myColor;
   }
/*}*/

}
```

FIG. 21

HelloWorldTranslator.java

```
/* XIS Tutorial standalone sequence step 3 data translator class. */

/*{*/
import com.xis.leif.im.AttributeGetRequest;
import com.xis.leif.im.AttributeSetRequest;
import com.xis.leif.im.Domain;
import com.xis.leif.im.Translator;
import com.xis.leif.im.FieldMetaData;
import com.xis.domains.display.DisplayDomain;
import com.xis.domains.movement.MovementDomain;
import java.awt.Color;

public class HelloWorldTranslator extends Translator {

   // the domains from which canned attribute metadata will be taken
2202 // NOTE, if an attribute appears in the methods below but its domain
   //      is NOT listed here, THE ATTRIBUTE WILL BE IGNORED BY XIS
   private static final Domain[] baseDomains = new Domain[] {
      DisplayDomain.getDomain(), MovementDomain.getDomain()
   };

2203 // store info about the fields, such as whether they are preferred or not
   private FieldMetaData[] fieldMetaDataArray;

2204 // Return the Domains that describe the Attributes.
   public Domain[] getBaseDomains() {
      return baseDomains;
   }

   // this method returns info on each field defined in the methods below
   public FieldMetaData[] getFieldMetaDataArray() {

      if (fieldMetaDataArray == null) {
         // initialize default metadata
       FieldMetaData dispname = new
                        FieldMetaData(DisplayDomain.displayName);
2206A    FieldMetaData pencolor = new
                        FieldMetaData(DisplayDomain.penColor);
         FieldMetaData speed = new
                        FieldMetaData(MovementDomain.speed);
       FieldMetaData course = new
                        FieldMetaData(MovementDomain.course);
```

FIG. 22A

Continuation of HelloWorldTranslator.java

```
        // attributes are visible ('preferred') by default; this
        // turns this off for the course attribute.
        course.setVisibility(false);
        // the order we put the attributes in here determines the order
        // they appear in tables or property sheets
        fieldMetaDataArray = new FieldMetaData[] {
          dispname, speed, course, pencolor
        };
    }

    return fieldMetaDataArray;
  }

  ////////////////////////////////////////////////////////////////
  // the following methods expose attributes of the HelloWorld class;
  // instead of calling the class methods directly, XIS will access
  // everything through this translator class
  ////////////////////////////////////////////////////////////////
  public String getDisplayName(AttributeGetRequest attributeGetRequest) {
    return ((HelloWorld)
        attributeGetRequest.getRawDataItem()).toString();
  }

  public Color getPenColor(AttributeGetRequest attributeGetRequest) {
    return ((HelloWorld)
        attributeGetRequest.getRawDataItem()).getMyColor();
  }

public void setPenColor(AttributeSetRequest attributeSetRequest,
                Color penColor) {
    HelloWorld helloWorld = (HelloWorld)
                    attributeSetRequest.getRawDataItem();
    Color oldPenColor = helloWorld.getMyColor();
    if (!penColor.equals(oldPenColor)) {
      helloWorld.setMyColor(penColor);
      // fire property change event to notify other XIS objects
      attributeSetRequest.getBaseDataItem().fireAttributeChanged(
        DisplayDomain.penColor, oldPenColor, penColor, true);
    }
  }

public double getSpeed(AttributeGetRequest attributeGetRequest) {
    return (double) ((HelloWorld)
        attributeGetRequest.getRawDataItem()).getValue();
  }
```

2206B

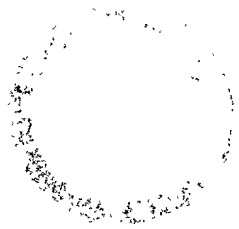
2210

FIG. 22B

Continuation of HelloWorldTranslator.java

```
    public void setSpeed(AttributeSetRequest attributeSetRequest,
                         double speed) {
      HelloWorld helloWorld = (HelloWorld)
                              attributeSetRequest.getRawDataItem();
2210  Double oldSpeed = new Double((double)helloWorld.getValue());
      if (oldSpeed.doubleValue() != speed) {
        helloWorld.setValue((int)speed);
        // fire property change event to notify other XIS objects
        attributeSetRequest.getBaseDataItem().fireAttributeChanged(
          MovementDomain.speed, oldSpeed, new Double(speed), true);
      }                                                              } 2212
    }

    // this is a dummy attribute to demonstrate field metadata
    public double getCourse(AttributeGetRequest attributeGetRequest) {   } 2208
      return (double) 0;
    }

  /*{*/
    // uncomment this to allow reflection to expose additional attributes
    // (see documentation under "Fooling Around")
    //   public HelloWorldTranslator() {
    //     introspectExcept(new String[] {"value", "myColor"});
    //   }
  /*}*/

  }
  /*}*/
```

FIG. 22C



TestHarness.java

```
/* XIS Tutorial standalone sequence step 3 XIS interfacing. */

import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import com.xis.propertysheet.PropertySheetInfoBean;
import com.xis.ui.UIBeanEvent;
import com.xis.ui.UIBeanAdapter;
import com.xis.leif.im.BaseInfoModel;
import jclass.chart.JCChart;
import com.xis.plot.PlotInfoBean;
import com.xis.plot.chartviews.LeifChartView;
/*{*/
import com.xis.leif.im.TranslatorRegistry;
import com.xis.leif.im.LeifDataItem;
import com.xis.leif.im.InfoModel;
import com.xis.leif.im.BaseInfoModel;
import com.xis.domains.movement.MovementDomain;
import com.xis.domains.movement.MovementDomainWrapper;
import com.xis.leif.im.LeifDataItemDelegator;
import java.lang.reflect.InvocationTargetException;
import com.xis.leif.im.UndefinedLeifAttributeException;
/*}*/

public class TestHarness {

/*{*/
   protected static LeifDataItem leifHello;

   static {
      // Register the translator for HelloWorld.  In fact this is really
      // only necessary when we have not followed the standard naming
      // convention (see docs), but it can't hurt.
      TranslatorRegistry.getTranslatorRegistry().registerObjectSchema(
         HelloWorld.class, HelloWorldTranslator.class);
   }
/*}*/

public static void main(String[] args) {

      // the plugin manager is only required for more complex applications
      // involving multiple components integrated at runtime
      BaseInfoModel.setStartingPlugInManager(false);

      HelloWorld hello = new HelloWorld();
```

2302

FIG. 23A

Continuation of TestHarness.java

```
// a property sheet infobean to display HelloWorld's attributes
PropertySheetInfoBean properties = new PropertySheetInfoBean();
properties.addRawDataItem(hello);

// add a listener for 'OK' or 'cancel', which generate 'close' events
properties.addUIBeanListener(
   new UIBeanAdapter() {
      public void closed(UIBeanEvent event) {
         System.exit(0);
      }
   }
);

// a top-level frame to hold our property sheet infobean
JFrame propertySheetFrame = new JFrame("HelloWorld Properties");
// add a listener for window closing
propertySheetFrame.addWindowListener(
   new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
         System.exit(0);
      }
   }
);

// stick the property Sheet bean in the frame and display it
propertySheetFrame.getContentPane().add(properties);
propertySheetFrame.pack();
propertySheetFrame.setVisible(true);
// now we create a plot infobean to plot HelloWorld's numeric attribute
PlotInfoBean plot = new PlotInfoBean();
plot.addRawDataItems(new Object[] { hello });
plot.setChartType(JCChart.BAR);
// the alternatives are SCATTER_PLOT, PLOT, AREA, PIE, CANDLE,
// and STACKING_BAR, though not all will make sense in this example

// We can set the attribute for initial display on the plot;
// if we do, this must consist of the attribute name preceded
// by the fully-qualified classname which ORIGINALLY DEFINES
// the attributeDescriptor -- i.e., using "HelloWorld.speed"
// here will NOT work!  If the descriptor is not defined in a
// domain or translator class, then it will have been defined
// dynamically through introspection when the first instance
// of the data item is dropped into an XIS InfoBean.
plot.setYAxisAttribute(
   "com.xis.domains.movement.MovementDomain.speed");
```

FIG. 23B

Continuation of TestHarness.java

```
        plot.setDynamicAdjustment(true);  // so axes track value magnitude
        plot.setBarChartAdjusting(true);  // needed in some cases for bar chart

        // a top-level frame as before to hold our plot infobean
        JFrame plotFrame = new JFrame("HelloWorld Plot");

        // stick the plot bean in and put it up
        plotFrame.getContentPane().add(plot);
        plotFrame.pack();
        plotFrame.setVisible(true);

/*{*/
        // create a leifDataItem version of hello and start a thread that
        // will increase it
        leifHello =
           BaseInfoModel.getBaseInfoModel().getLeifDataItem(hello);
        new Accelerate();
/*}*/
    } // main

}

/*{*/
// thread to update the speed attribute on the leifHello instance we created
class Accelerate extends Thread {

        public Accelerate() {
           super("Accelerator Thread");
           start();
        }

        public void run() {

           // wrap the LeifDataItem leifHello in a convenience wrapper that
           // gives access to attributes within that domain, if they exist
           MovementDomainWrapper helloMovementWrapper =
              MovementDomain.takeWrapper(TestHarness.leifHello);
```

FIG. 23C

Continuation of TestHarness.java

```
        while (true) {

            // sleep for 0.5 seconds, then...
            try {
               sleep(500);
            } catch (InterruptedException e) {
               System.exit(1);
            }

            // ..update the speed attribute
            try {
               helloMovementWrapper.setSpeed(
                        helloMovementWrapper.getSpeed()+1);
            } catch (UndefinedLeifAttributeException ulae) {
               // exception if this data item doesn't have this attribute
               System.exit(1); // usually we would do something better
            } catch (InvocationTargetException ite) {
               // sweep up any exception tossed by the underlying raw item
               System.exit(1); // usually we would do something better
            }
         } // while
      } // run()
};
/*}*/
```

FIG. 23D

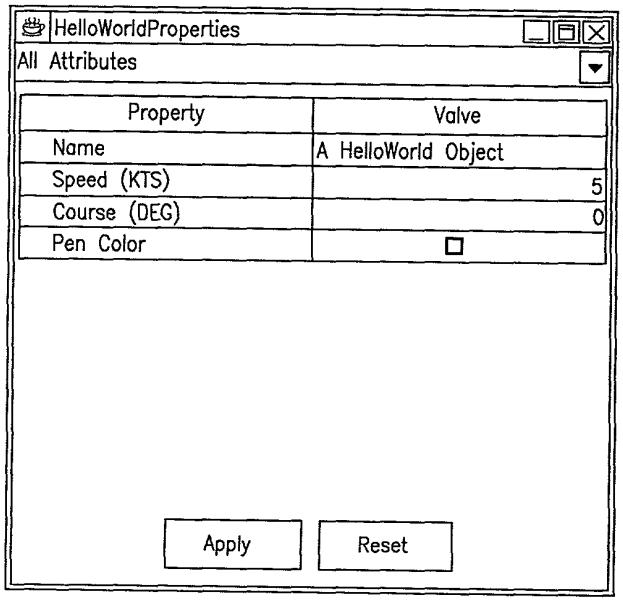


FIG. 24A

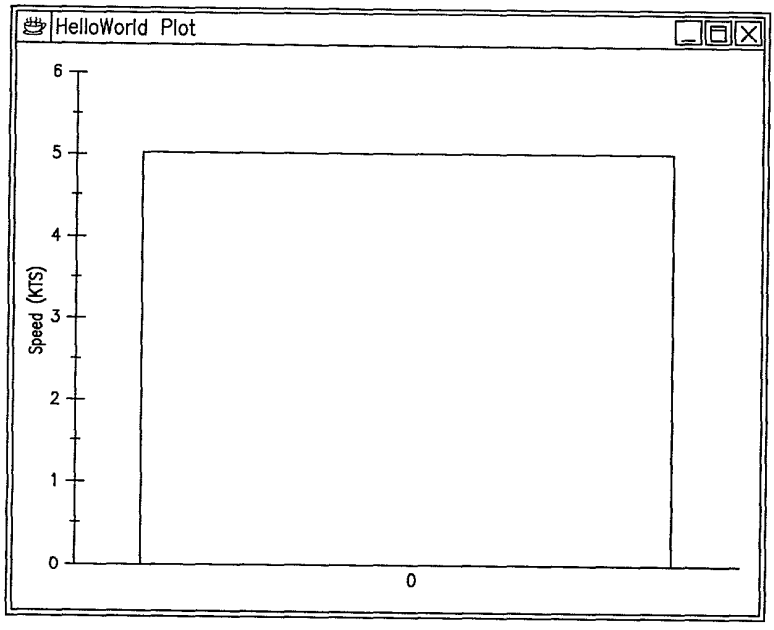


FIG. 24B

HelloWorld.java

```
/* XIS Tutorial standalone sequence step 4 data class. */

import java.awt.Color;
/*{*/
import com.xis.leif.im.FieldMetaData;
import com.xis.domains.display.DisplayDomain;
import com.xis.domains.movement.MovementDomain;
import com.xis.leif.im.Domain;
import com.xis.leif.im.AttributeGetRequest;
import com.xis.leif.im.AttributeSetRequest;
import com.xis.leif.im.AttributeDescriptor;
import java.beans.PropertyChangeSupport;
import java.beans.PropertyChangeListener;
/*}*/

public class HelloWorld {

   private int value = 1;
   private Color myColor = Color.green;

/*{*/
   public static AttributeDescriptor getDisplayNameDescriptor() {
      return DisplayDomain.displayName;
   }
   public static AttributeDescriptor getSpeedDescriptor() {
      return MovementDomain.speed;
   }
   public static AttributeDescriptor getPenColorDescriptor() {
      return DisplayDomain.penColor;
   }
/*}*/

/*{*/   // this property change support code as in step 2  /*}*/
   // this member class helps distribute property change events within XIS
   private PropertyChangeSupport propertyChangeSupport =
                              new PropertyChangeSupport(this);

// two aux methods to let other XIS objects pay attention to this one
   public void addPropertyChangeListener(PropertyChangeListener l) {
      propertyChangeSupport.addPropertyChangeListener(l);
   }
   public void removePropertyChangeListener(PropertyChangeListener l) {
      propertyChangeSupport.removePropertyChangeListener(l);
   }
```

2502

2504

FIG. 25A

Continuation of HelloWorld.java

```
    public String toString() {
        return "A HelloWorld Object";
    }

    public String getGreeting() {
        return "Hello World!";
    }

    public int getID() {
        return 5;
    }

    public int getValue() {
        return value;
    }

/*{*/ // this function as in step 2 /*}*/
    public void setValue(int value) {
        // we only want to update and fire property change if really changes
        if (this.value != value) {
            int oldValue = this.value;
            this.value = value;
            // fire property change event to notify other XIS objects
            propertyChangeSupport.firePropertyChange("value", oldValue, value);
        }
    }

/*{*/
// "myColor"-related methods changed to expose "penColor" instead

    public Color getPenColor() {
        return myColor;
    }

    public void setPenColor(Color penColor) {
        // we only want to update and fire property change if really changes
        if (penColor != this.myColor) {
            Color oldPenColor = this.myColor;
            this.myColor = penColor;
            // fire property change event to notify other XIS objects
            propertyChangeSupport.firePropertyChange("penColor",
                                        oldPenColor, penColor);
        }
    }
```

FIG. 25B

Continuation of HelloWorld.java

```
    // expose toString() return under a new name
    public String getDisplayName() {
       return toString();
    }
 // expose "value" under a new name
    public double getSpeed() {
       return (double) getValue();
    }

 public void setSpeed(double speed) {
       // we only want to update and fire property change if really changes
       Double oldSpeed = new Double((double)this.getValue());
       if (speed != oldSpeed.doubleValue()) {
          setValue((int)speed);
          // fire property change event to notify other XIS objects
          propertyChangeSupport.firePropertyChange("speed", oldSpeed,
                                        new Double(speed));
       }
    }
 /*}*/

 /*{*/
    // store info about the fields, such as whether they are preferred or not
    private static FieldMetaData[] fieldMetaDataArray;

    // this method returns info on each field defined in the methods below
    public static FieldMetaData[] getFieldMetaDataArray() {

       if (fieldMetaDataArray == null) {
          // initialize default metadata
          FieldMetaData dispname = new
                        FieldMetaData(DisplayDomain.displayName);
          FieldMetaData pencolor = new
                        FieldMetaData(DisplayDomain.penColor);
          FieldMetaData speed = new
                        FieldMetaData(MovementDomain.speed);
          // could customize the field metadata here
          fieldMetaDataArray = new FieldMetaData[] {
             dispname, speed, pencolor
          };
       }
       return fieldMetaDataArray;
    }
 /*}*/
 }
```

2506

FIG. 25C

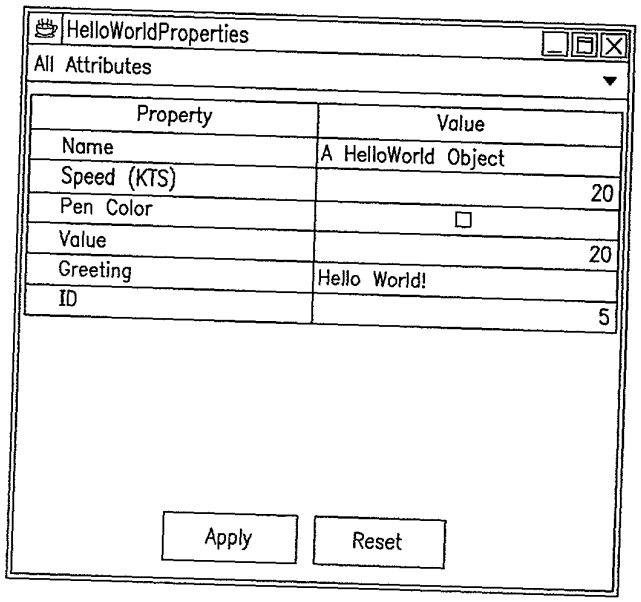

HelloWorldProperties
All Attributes
Property
Value
Name
A HelloWorld Object
Speed (KTS)
20
Pen Color
Value
20
Greeting
Hello World!
ID
5
Apply
Reset

FIG. 26

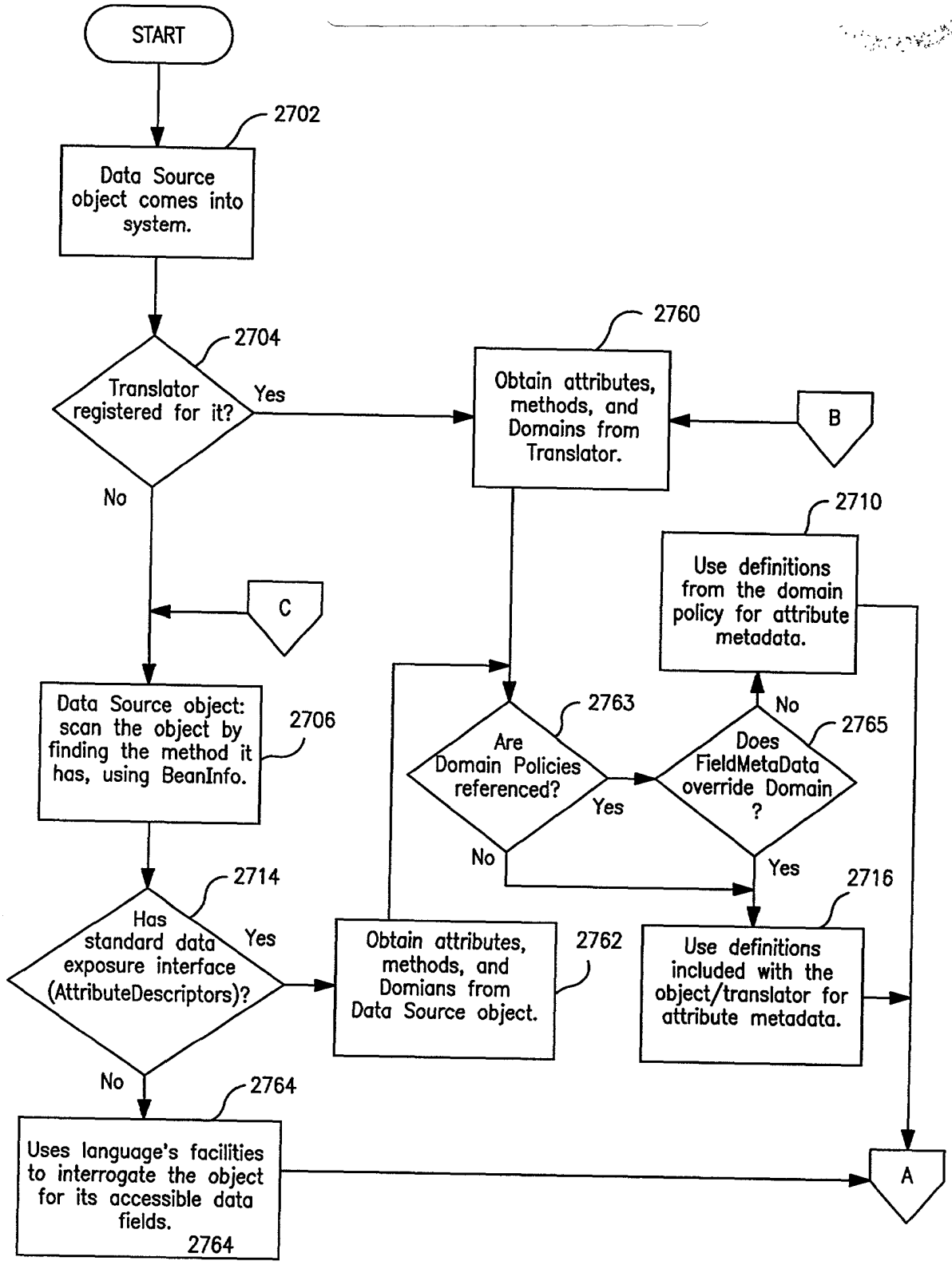

START
2702
Data Source object comes into system.
2704
Translator registered for it?
Yes
No
2760
Obtain attributes, methods, and Domains from Translator.
B
C
2710
Use definitions from the domain policy for attribute metadata.
Data Source object: scan the object by finding the method it has, using BeanInfo.
2706
2763
Are Domain Policies referenced?
Yes
No
2765
Does FieldMetaData override Domain ?
No
Yes
2714
Has standard data exposure interface (AttributeDescriptors)?
Yes
No
Obtain attributes, methods, and Domians from Data Source object.
2762
2716
Use definitions included with the object/translator for attribute metadata.
2764
Uses language's facilities to interrogate the object for its accessible data fields.
2764
A

FIG. 27A

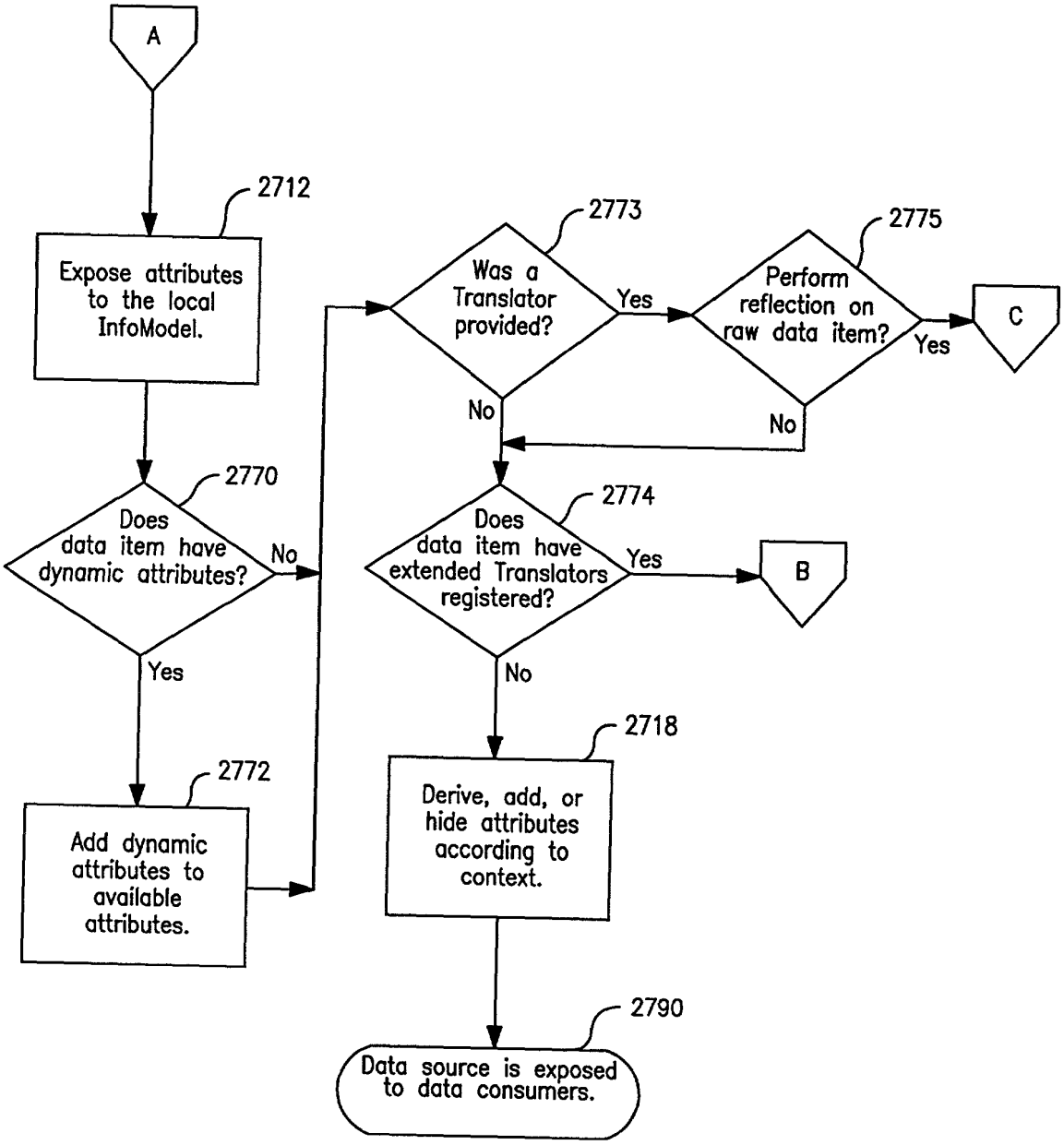

A
2712
Expose attributes to the local InfoModel.
2770
Does data item have dynamic attributes?
No
Yes
2772
Add dynamic attributes to available attributes.
2773
Was a Translator provided?
Yes
No
2775
Perform reflection on raw data item?
Yes
No
C
2774
Does data item have extended Translators registered?
Yes
B
No
2718
Derive, add, or hide attributes according to context.
2790
Data source is exposed to data consumers.

FIG. 27B

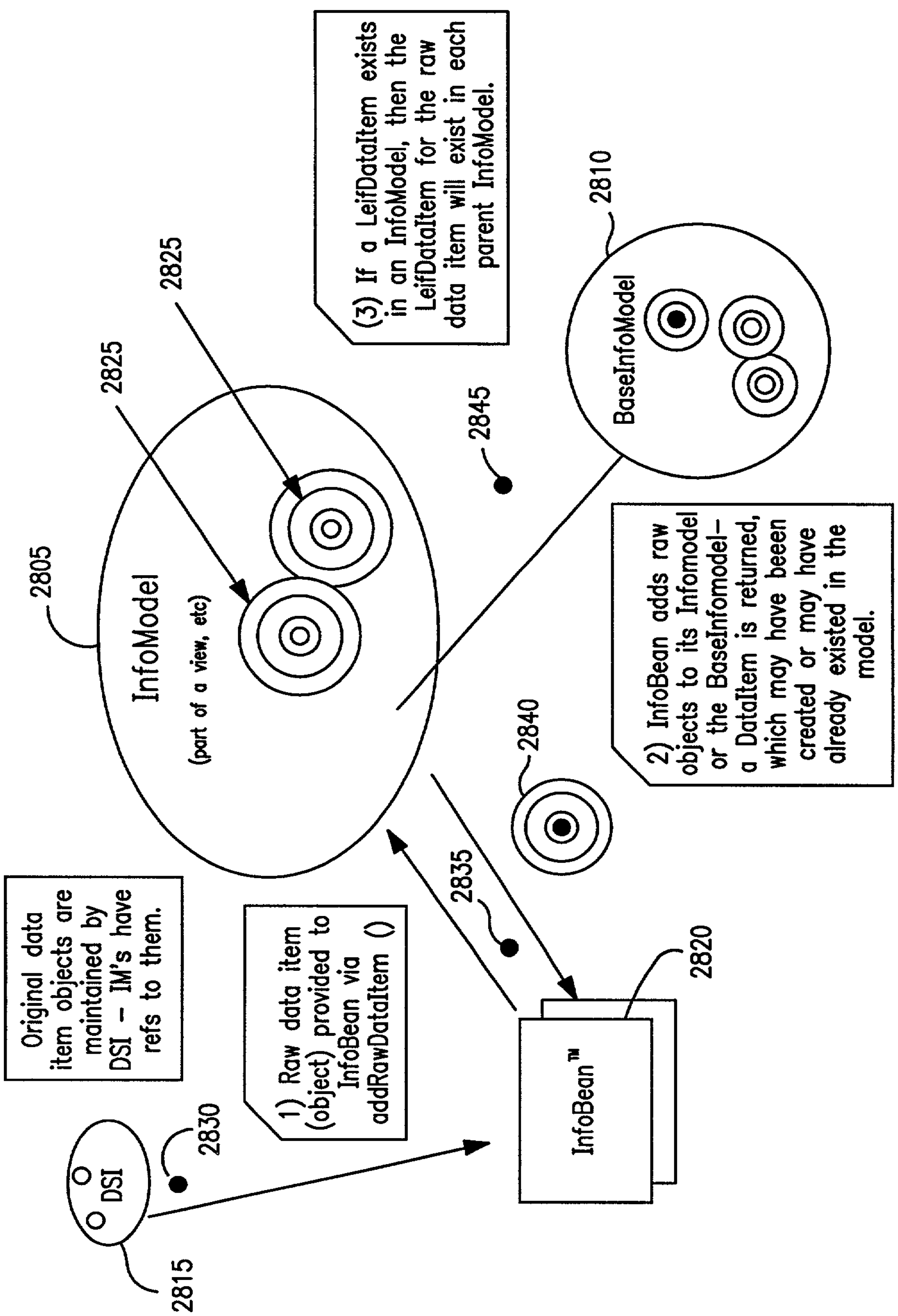

Original data item objects are maintained by DSI – IM's have refs to them.
1) Raw data item (object) provided to InfoBean via addRawDataItem ()
DSI
2815
2830
InfoBean™
2820
2835
InfoModel
(part of a view, etc)
2805
2825
2825
2840
2845
2) InfoBean adds raw objects to its Infomodel or the BaseInfomodel– a DataItem is returned, which may have beeen created or may have already existed in the model.
BaseInfoModel
2810
(3) If a LeifDataItem exists in an InfoModel, then the LeifDataItem for the raw data item will exist in each parent InfoModel.

FIG. 28

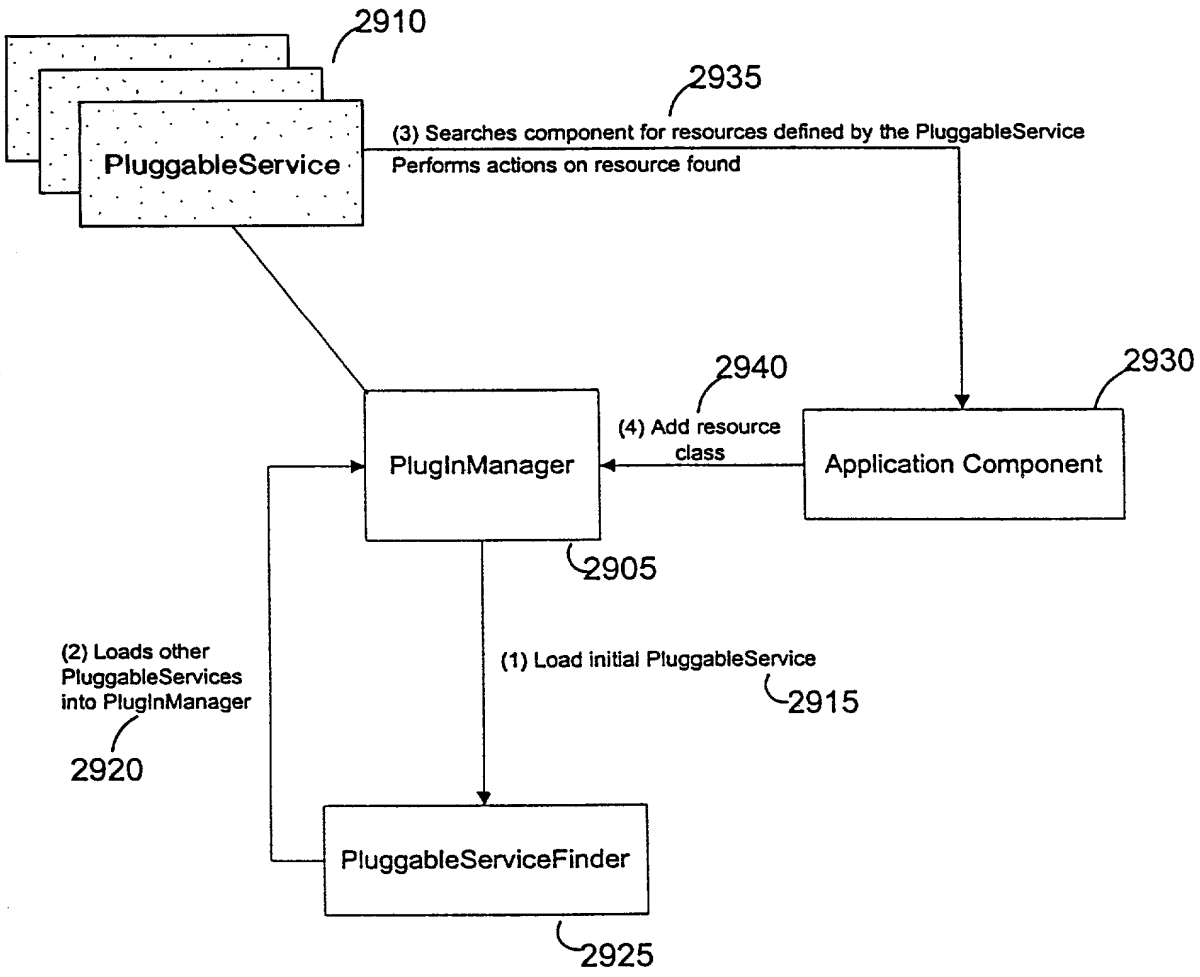

2910
2935
PluggableService
(3) Searches component for resources defined by the PluggableService
Performs actions on resource found
2940
2930
PlugInManager
(4) Add resource class
Application Component
2905
(2) Loads other PluggableServices into PlugInManager
2920
(1) Load initial PluggableService
2915
PluggableServiceFinder
2925

FIG. 29

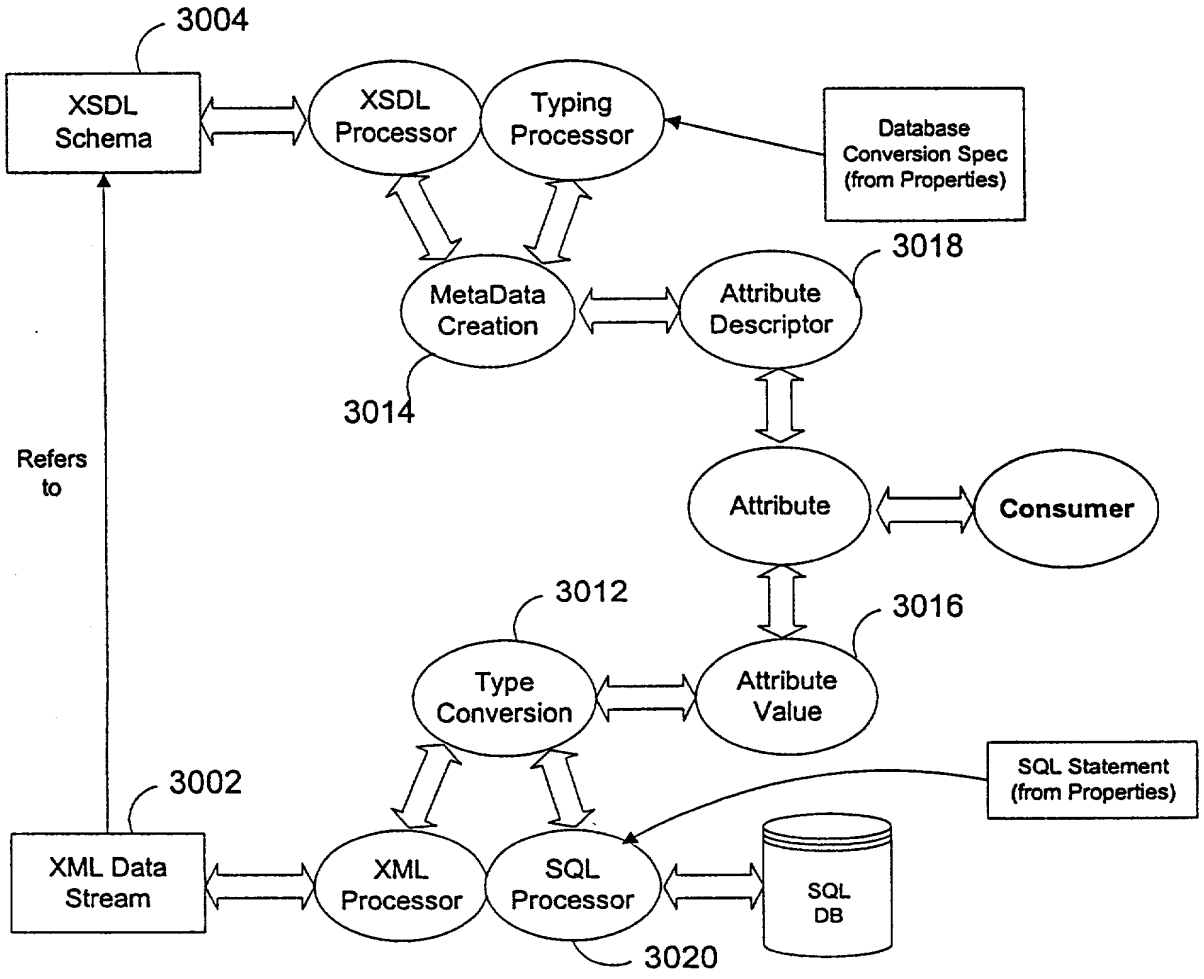

3004
XSDL Schema
XSDL Processor
Typing Processor
Database Conversion Spec (from Properties)
3018
MetaData Creation
Attribute Descriptor
3014
Refers to
Attribute
Consumer
3012
3016
Type Conversion
Attribute Value
SQL Statement (from Properties)
3002
XML Data Stream
XML Processor
SQL Processor
SQL DB
3020

FIG. 30

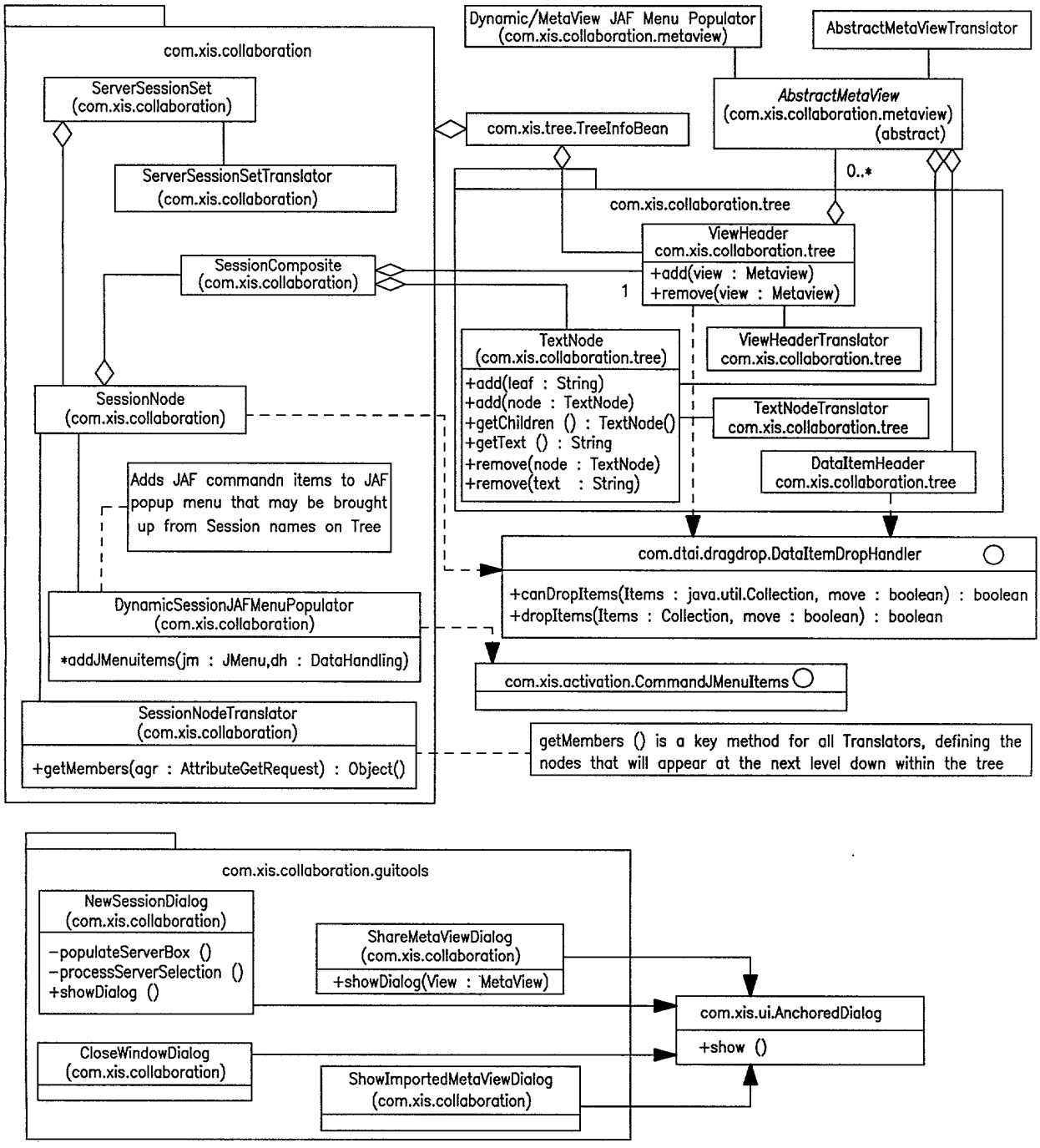

com.xis.collaboration
ServerSessionSet
(com.xis.collaboration)
ServerSessionSetTranslator
(com.xis.collaboration)
SessionComposite
(com.xis.collaboration)
SessionNode
(com.xis.collaboration)
Adds JAF commandn items to JAF popup menu that may be brought up from Session names on Tree
DynamicSessionJAFMenuPopulator
(com.xis.collaboration)
*addJMenuitems(jm : JMenu,dh : DataHandling)
SessionNodeTranslator
(com.xis.collaboration)
+getMembers(agr : AttributeGetRequest) : Object()
Dynamic/MetaView JAF Menu Populator
(com.xis.collaboration.metaview)
AbstractMetaViewTranslator
AbstractMetaView
(com.xis.collaboration.metaview)
(abstract)
com.xis.tree.TreeInfoBean
0..*
com.xis.collaboration.tree
ViewHeader
com.xis.collaboration.tree
+add(view : Metaview)
+remove(view : Metaview)
1
TextNode
(com.xis.collaboration.tree)
+add(leaf : String)
+add(node : TextNode)
+getChildren () : TextNode()
+getText () : String
+remove(node : TextNode)
+remove(text : String)
ViewHeaderTranslator
com.xis.collaboration.tree
TextNodeTranslator
com.xis.collaboration.tree
DataItemHeader
com.xis.collaboration.tree
com.dtai.dragdrop.DataItemDropHandler
+canDropItems(Items : java.util.Collection, move : boolean) : boolean
+dropItems(Items : Collection, move : boolean) : boolean
com.xis.activation.CommandJMenuItems
getMembers () is a key method for all Translators, defining the nodes that will appear at the next level down within the tree
com.xis.collaboration.guitools
NewSessionDialog
(com.xis.collaboration)
-populateServerBox ()
-processServerSelection ()
+showDialog ()
ShareMetaViewDialog
(com.xis.collaboration)
+showDialog(View : MetaView)
CloseWindowDialog
(com.xis.collaboration)
ShowImportedMetaViewDialog
(com.xis.collaboration)
com.xis.ui.AnchoredDialog
+show ()

FIG. 31

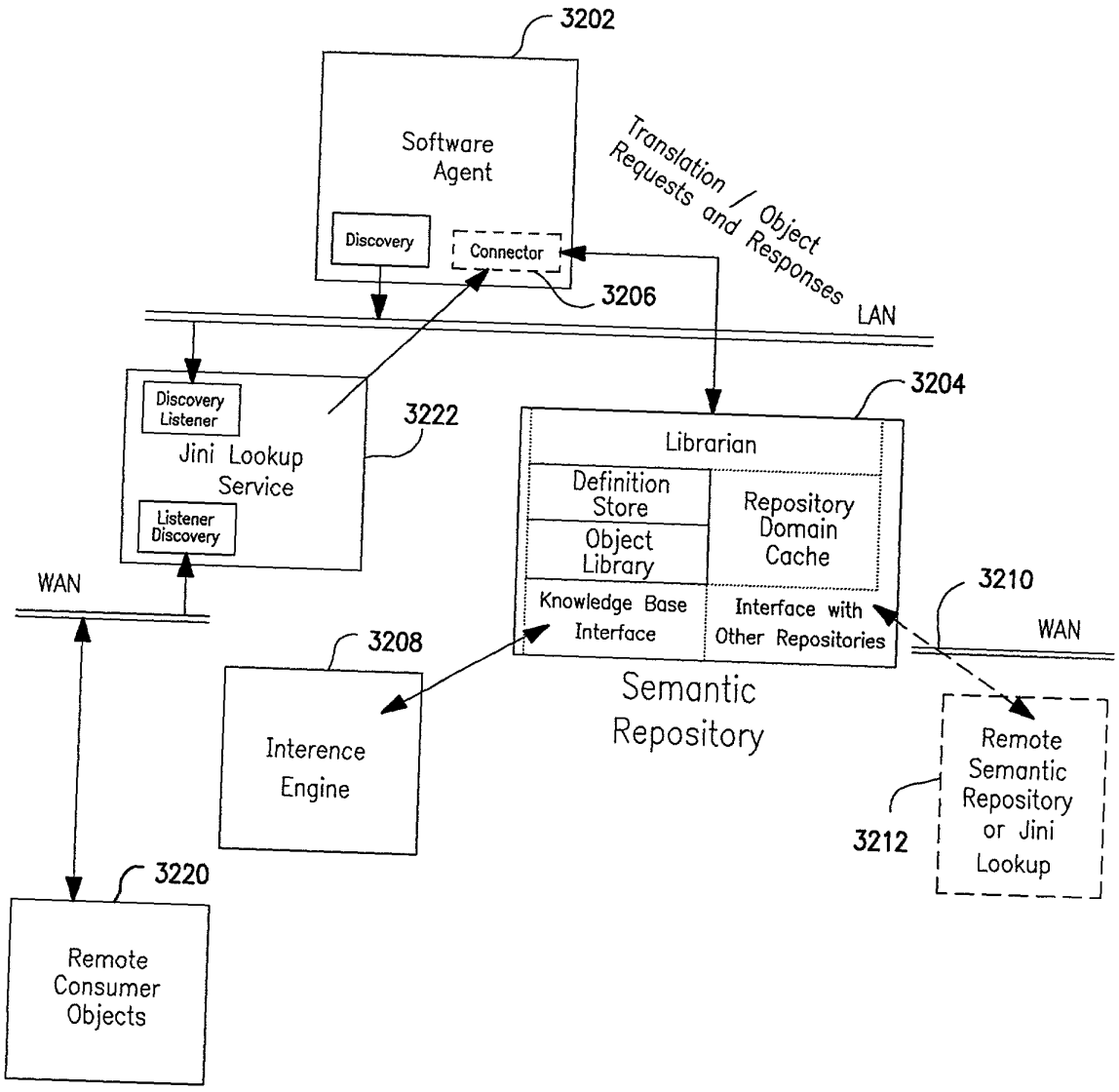

3202
Software Agent
Discovery
Connector
3206
Translation / Object Requests and Responses
LAN
Discovery Listener
Jini Lookup Service
Listener Discovery
3222
3204
Librarian
Definition Store
Object Library
Repository Domain Cache
Knowledge Base Interface
Interface with Other Repositories
Semantic Repository
3210
WAN
WAN
3208
Interence Engine
3212
Remote Semantic Repository or Jini Lookup
3220
Remote Consumer Objects

FIG. 32

Class ContentInfoBean

java.lang.Object
  +--java.awt.Component
    +--java.awt.Container
      +--javax.swing.JComponent
        +--javax.swing.JPanel
          +--com.xis.ui.AbstractUIBean
            +--com.xis.leif.infobeans.DataItemsSinkUIBean
              +--com.xis.infobeans.content.ContentInfoBean

All Implemented Interfaces:
Accessible, BeanContextChildOwner, BeanContextChildOwnerDelegator, BeanContextProxy, BeanContextServicesOwnerDelegator, ClipboardUser, DataItemSink, ImageObserver, MenuContainer, Serializable, StateSavable, UIBean

---

public class ContentInfoBean
extends DataItemSinkUIBean
implements ClipboardUser

The ContentInfoBean class is a visual component that displays the contents of a raw data item. If no contents are available, it defaults to a split pane containing the JAF menu and the PropertySheet of the raw data item. The contents may have be multipart, and may be text, html, rich text, or an image. Multimedia support will soon be added.

Author:
Jaime Garcia, Polexis, Inc
See Also:
Serialized Form

Inner classes inherited from class javax.swing.JPanel
JPanel.AccessibleJPanel

Inner classes inherited from class javax.swing.JComponent
JComponent.AcessibleJComponent

Inner classes inherited from class java.awt.Container
Container.AccessibleAWTContainer

FIG. 33A

Inner classes inherited from class java.awt.Component
Component.AccessibleAWTComponent

Field Summary

static TypedResourceBundle	RESOURCES localized resources for this view object.

Fields inherited from class com.xis.ui.AbstractUIBean
uibeanListener

Fields inherited from class javax.swing.JComponent
accessibleContext, ListenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interaface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

ContentInfoBean ()
Default constructor that creates an empty ContentInfoBean.

Method Summary

void	addRawDataItem (Object rawDataItem) Load the raw data item into the ContentInfoBean.
void	addRawDataItems (Object [] rawDataItems) Add the raw data items in the array.
boolean	canClear () Return true if the ContentInfoBean has an object and it is selected
boolean	canClear (Object [] items) Return true if the specified items can be cleared.
boolean	canCopy () Return true if the ContentInfoBean has an object and it is selected
boolean	canCopy (Object [] items) Return true if the specified items can be copied.
boolean	canCut () Return true if the ContentInfoBean has an object and it is selected
boolean	canCut (Object [] items) Return true if the specified items can be cut.
boolean	canPaste () Return true if the ContentInfoBean can paste new objects, false if not.
boolean	canSelectAll () Return true if the ContentInfoBean can select all objects.

FIG. 33B

boolean	canSelectNone () Return true if the ContentInfoBean can un–select all objects.
void	clear () Notify the ContentInfoBean to remove the current raw data item only if it is selected.
void	clear (Object [] items) Clears the given items.
void	clearAll () Removes the currently loaded object.
boolean	contains (Object [] items) Return true if this ContentInfoBean contains *all* the objects of the given array.
boolean	containsComponent (Component component) Check if the given component is contained by this InfoBean
void	copy (Clipboard clipboard) Called to invoke this ContentInfoBean's copy action, which is to copy all selected data to the Clipboard.
void	copy (Clipboard clipboard, Object [] items) Copies the given items into the Clipboard.
protected void	createContent () Method called when there is no content to display for a raw data item.
void	cut (Clipboard clipboard) Cut selected items from the ContentInfoBean and post them into Clipboard.
void	cut (Clipboard clipboard, Object [] items) Cut the given items from the ContentInfoBean and post them into the given Clipboard only if they occur in the ContentInfoBean.
protected Container	getContainerForContent (int index) Get a container with the contents of the content object at the given index, or null if the content type is not supported.
Object	getContents () Fetch the currently loaded raw data item
JAFAndPropertyComponent	getJAFAndPropertyComponent () Get the JAFAndProperty component used by the ContentInfoBean to display the contents for raw data items that have nothing else to display.
JMenu	getLeifDataItemMenu (LeifDataItem dataItem, boolean showCutPastItems) Return the data item menu for a LeifDataItem (usually the selected LeifDataItem).
TypedResourceBundle	getResources () Return the ResourceBundle for this ContentInfoBean.
Object ()	getSelectedObjects () Get an array of selcted objects.
Void	infoModelChanged () Messaged to indicate an InfoModel change for this InfoBean or one or more of its LeifDataItems.
boolean	isCreatingContent () Check whether default content creation is set.
boolean	isDragEnabled () Return true if the default Drag support is enabled.
boolean	isDropEnabled () Return true if the default Drop support is enabled.

FIG. 33C

boolean	isSelected () Check the selected state of the content object, if there is currently one loaded.
boolean	isXISNotifying () Check whether the ContentInfoBean is updating based on XIS events and is notifying XIS of raw data item attribute changes
void	paste (Clipboard clipboard) Paste the data Objects from the given clipboard.
void	removeAllRawDataItems () Remove all of the raw items that are currently loaded
void	removeRawDataItem (Object rawDataItem) Remove the given raw data item if it is the currently loaded raw data item
void	removeRawDataItems (Object [] rawDataItems) Remove the given raw data items in the array.
void	selectAll () Set the selection state of the object to true.
void	selectNone () Set the selection state of the object to false.
void	setCreateContent (boolean create) Set whether content should be created for objects that do not have any displayable content, via a JAFAndPropertyComponent.
void	setDragEnabled (boolean enabledrag) Set the stratus of the default Drag support.
void	setDragOwnerProxy (DragOwner dragProxy) Set the DragOwner "proxy" for this InfoBean.
void	setDropEnabled (boolean enabledrop) Set the status of the default Drop Support.
void	setDropOwnerProxy (DropOwner dropProxy) Set a DropOwner "proxy" for this InfoBean.
void	setSelection (boolean selected) Set a selection state of the content object
void	setXISNotifying (boolean notify) Set whether ContentInfoBean should update based on XIS events and should notify XIS of raw data item attribute changes

Methods inherited from class com.xis.leif.infobeans.DataItemSinkUIBean
addJAFPopulator, addRawDataItemAsGroup, addService, close, createBeanContextServicesOwnerDelegator, dispose, getBeanContextProxy, getBeanContextServices, getEzContext, getInfoModel, getJAFPopulators, getLeifDataItemMenu, getLeifDataItemMenu, getLeifDataItemMenu, getMenuBar, getOwnedBeanContextChild, getService, getService, getService, getStatusBars, getToolBars, getUIs, initializeBeanContextResources, intializeBeanContextServices, invalidateInfoModel, isDropPastEnabled, isHandlingClipboardOperations, releaseBeanContextResources, releaseBeanContextServices, revokeAnchoredDialogProvider, revokeFrameProvider, revokeService, setDropPasteEnabled, setHandlingClipboardOperations, validatePendingSetBeanContext

Methods inherited from class com.xis.ui.AbstractUIBean
addUIBeanListener, finalize, getShortTitle, getTitle, getUIComponents, isActive, isClosed, isCloseOK, processUIBeanEvent, removeUIBeanListener, restoreState, saveState, setActive, setShortTitle, setTitle

FIG. 33D

Methods inherited from class javax.swing.JPanel
getAccessibleContext, getUIClassID, paramString, updateUI

Methods inherited from class javax.swing.JComponent
addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangerListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionmap, getAlignmentX, getAlignmentY, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getGraphics, getHeight, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, GetMinimumSize, getNextFocusableComponent, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getVerifyInputWhenFocusTarget, getVisibleRect, getWidth, getX, getY, grabFocus, hasFocus, hide, isDoubleBuffered, isFocusCycleRoot, isFocusTraversable, isLightweightComponent, isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processFocusEvent, processKeyBinding, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setDebugGraphicsOptions, setDoubleBufffered, setEnabled, setFont, setForeground, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpapue, setPreferredSize, setRequestFocusEnabled, setToolTipText, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getLayout, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setLayout, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionlistener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getInputContext, getInputMethodRequests, getLocale, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getToolKit, getTreeLock, gotFocus, handleEvent, imageUpdate, inside, isDisplayable, isEnabled, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus

FIG. 33E

Methods inherited from calss java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface com.xis.ui.ClipboardUser
isHandlingClipboardOperations, setHandlingClipboardOperations

Methods inherited from interface com.xis.ui.UIBean
addPropertyChangeListener, removePropertyChangeListener

Field Detail

RESOURCES

public static final TypedResourceBundle RESOURCES

localized resources for this view object.

Constructor Detail

ContentInfoBean

public ContentInfoBean ()

Default constructor that creates an empty ContentInfoBean.

Method Detail

getContents

public Object getContents ()

Fetch the curently loaded raw data item
Returns:
the currently loaded object, or null if no object is loaded

---

setXISNotifying

public void setXISNotifying (boolean notify)

Set whether the ContentInfoBean should update based on XIS events and should notify XIS of raw data item attribute changes
Parameters:
notify – if true then notify XIS, else do not

FIG. 33F

isXISNotifying

```
public boolean isXISNotifying()
```

Check whether the ContentInfoBean is updating based on XIS events and is notifying XIS of raw data item attribute changes
Returns:
true if it is notifying XIS, else false

---

removeAllRawDataItems

```
public void removeAllRawDataItems()
```

Remove all of the raw data itmes that are currently loaded

---

getResources

```
public TypedResourceBundle getResources()
```

Return the ResourceBundle for this ContentInfoBean..
Overrides:
getResources in class DataItemSinkUIBean
Returns:
the statically sourced ResourceBundle.

---

infoModelChanged

```
public void infoModelChanged()
```

Messaged to indicate an InfoModel change for this InfoBean or one or more of its LeifDataItems. This should reload all currently loaded data items to pick up InfoModel changes.
Overrides:
infoModelChanged in class DataItemSinkUIBean

---

addRawDataItems

```
public void addRawDataItems(Object() rawDataItems)
```

Add the raw data items in the array. Will only add if the array has only one object and the object is not the currently loaded rawDataItem.
Parameters:
rawDataItems—the array of objects to be added

---

addRawDataItem

```
public void addRawDataItem(Object rawDataItem)
```

FIG. 33G

Load the raw data item into the ContentInofBean. If the raw data item supports the ContentDomain and has content of type that is currently supported then it will be displayed. Otherwise, if content creation is set to true then default content will be created, using a JAFAndPropertyComponent
Overrides:
addRawDataItem in class DataItemSinkUIBean
Parameters:
rawDataItem–the raw object to display

---

createContent

```
Protected void createContent ()
```

Method called when there is no content to display for a raw data item. The default behavior displays a JAFAndPropertyComponent of the object, but subclasses may wish to display something else instead.

---

isCreatingContent

```
public boolean isCreatingContent()
```

Check whether default content creation is set. If so then a JAFAndPropertyComponent will be created for items that have no displayable content.
Returns:
true if default content is created, else false

---

setCreateContent

```
public void setCreateContent(boolean create)
```

Set whether content should be created for objects that do not have any displayable content, via a JAFAndPropertyComponent.
Parameters:
create–if true, create content, else do not

---

removeRawDataItems

```
public void removeRawDataItems(Object() rawDataItems)
```

Remove the raw data items in the array. Will only remove the object that is currently loaded if it is contained in the array
Parameters:
rawDataItems–the array of objects to be removed

---

removeRawDataItem

```
public void removeRawDataItem(Object rawDataItem)
```

FIG. 33H

Remove the given raw data item if it is the currently loaded raw data item
Overrides:
removeRawDataItem in class DataItemSinkUIBean
Parameters:
rawDataItem—the object to remove

---

getJAFAndPropertyComponent

```
public JAFAndPropertyComponent get JAFAndPropertyComponent()
```

Get the JAFAndProperty component used by the ContentInfoBean to display the contents for raw data items that have nothing else to display.
Returns:
the current JAFAndPropertyComponent being used

---

getContainerForContent

```
protected Container getContainerForContent(int index)
```

Get a container with the contents of the content object at the given index, or null if the content type is not supported. Subclass this if you need support for a type that is not already supported.
Parameters:
index—the index of in the object
Returns:
a Container with the contents at the given index

---

getSelectedObjects

```
public Object() getSelectedObjects()
```

Get an array of selected objects. This will return an empty array if there are no selected objects. If the raw data item is selected it will return an array of size 1 with the raw data item inside
Returns:
an array of selected objects

---

isSelected

```
public boolean isSelected()
```

Check the selected state of the content object, if there is currently one loaded. If there is none, return false.
Returns:
true if there is an object and it is selected

---

selectAll

```
public void selectAll()
```

FIG. 33I

Set the selection state of the object to true.
Specified by:
selectAll in interface ClipboardUser
Overrides:
selectAll in class DataItemSinkUIBean

---

selectNone

```
public void selectNone()
```

Set the selection state of the object to false.

---

setSelection

```
public void setSelection(boolean selected)
```

Set the selection state of the content object
Parameters:
the–new selection state

---

canClear

```
public boolean canClear()
```

Return true if the ContentInfoBean has an object and it is selected
Specified by:
canClear in interface ClipboardUser
Overrides:
canClear in class DataItemSinkUIBean
Returns:
true if the raw data item is selected
See Also:
getSelectedObjects()

---

canClear

```
public boolean canClear(Object() items)
```

Return true if the specified items can be cleared. If the item is not in the ContentInfoBean return false. If there is more than one item return false.
Specified by:
canClear in interface ClipboardUser
Overrides:
canClear in class DataItemSinkUIBean
Parameters:
items–the Array of items to be cut.
Returns:
true if all of the items are present, otherwise false
See Also:

FIG. 33J

contains(Object [])

---

canCopy

```
public boolean canCopy()
```

Return true if the ContentInfoBean has an object and it is selected
Specified by:
canCopy in interface ClipboardUser
Overrides:
canCopy in class DataItemSinkUIBean
Returns:
true if the raw data item is selected
See Also:
canClear ()

---

canCopy

```
public boolean canCopy(Object[] items)
```

Return true if the specified items can be copied. If the item is not in the ContentInfoBean return false. If there is more than one item return false.
Specified by:
canCopy in interface ClipboardUser
Overrides:
canCopy in class DataItemSinkUIBean
Parameters:
items–the Array of itmes to be cut.
Returns:
true if all of the items are present, otherwise false
See Also:
canClear(Object [])

---

canCut

```
public boolean canCut()
```

Return true if the ContentInfoBean has an object and it is selected
Specified by:
canCut in interface ClipboardUser
Overrides:
canCut in class DataItemSinkUIBean
Returns:
true if the raw data item is selected
See Also:
canClear()

---

canCut

```
public boolean canCut(Object [] items)
```

FIG. 33K

Return true if the specified items can be cut. If the item is not in the ContentInfoBean return false. If there is more than one item return false.
Specified by:
canCut in interface ClipboardUser
Overrides:
canCut in class DataItemSinkUIBean
Parameters:
items—the Array of items to be cut.
Returns:
true if all of the items are present, otherwise false
See Also:
canClear(Object [])

---

canPaste

```
public boolean canPaste()
```

Return true if the ContentInfoBean can paste new objects, false if not.
Specified by:
canPaste in interface ClipboardUser
Overrides:
canPaste class DataItemSinkUIBean
Returns:
default is always true for the ContentInfoBean.

---

canSelectAll

```
public boolean canSelectAll()
```

Return true if the ContentInfoBean can select all objects.
Specified by:
canSelectAll in interface ClipboardUser
Overrides:
canSelectAll in class DataItemSinkUIBean
Returns:
returns true if there is an object and it is not selected

---

canSelectNone

```
public boolean canSelectNone()
```

Return true if the ContentInfoBean can un—select all objects.
Returns:
returns true if the raw data item is selected

---

clear

```
public void clear()
```

FIG. 33L

Notify the ContentInfoBean to remove the current raw data item only if it is selected.
Specified by:
clear in interface ClipboardUser
Overrides:
clear in class DataItemSinkUIBean

---

clear

```
public void clear(Object [] items)
```

Clears the given items. These items only get cleared if they actually occur in the ContentInfoBean.
Specified by:
clear in interface ClipboardUser
Overrides:
clear in class DataItemSinkUIBean
Parameters:
items—the items to cleared

---

clearAll

```
public void clearAll()
```

Removes the currently loaded object.

---

copy

```
public void copy(Clipboard clipboard)
```

Called to invoke theis ContentInfoBean's copy action, which is to copy all selected data to the Clipboard.
Specified by:
copy in interface ClipboardUser
Overrides:
copy in class DataItemSinkUIBean
Parameters:
clipboard—the Clipboard object that gets posted to. The actual items posted are contained in a LeifTransferable.

---

copy

```
public void copy(Clipboard clipboard,
                 Object [] items)
```

Copies the given items into the Clipboard.
Specified by:
copy in interface ClipboardUser
Overrides:
copy in class DataItemSinkUIBean
Parameters:
clipboard—the Clipboard object that gets posted to. The actual items posted are contained in a

FIG. 33M

FIG. 33N

LeifTransferable.
items – the array of Object items to copied.

---

cut

public void cut(Clipboard clipboard)

Cut selected items from the ContentInfoBean and post them into Clipboard.
Specified by:
cut in interface ClipboardUser
Overrides:
cut in class DataItemSinkUIBean
Parameters:
clipboard–the Clipboard object that gets posted to. The actual items posted are contained in a LeifTransferable.

---

cut

public void cut(Clipboard clipboard,
Object [] items)

Cut the given items from the ContentInfoBean and post them into the given Clipboard only if they occur in the ContentInfoBean.
Specified by:
cut in interface ClipboardUser
Overrides:
cut in class DataItemSinkUIBean
Parameters:
clipboard–the Clipboard object that gets posted to. The actual items posted are contained in a LeifTransferable.
items – the array of Object item to cut and posted.

---

paste

public void paste (Clipboard clipboard)

Paste the data Objects from the given clipboard. This retrieves the clipboard contents and does a simple add.
Specified by:
paste in interface ClipboardUser
Overrides:
paste in class DataItemSinkUIBean
Parameters:
clipboard – the Clipboard that contains the objects.
See also:
addRawDataItems (Object [])

---

contains

public final boolean contains (Object [] items)

FIG. 330

Return true if this ContentInfoBean contains *all* the objects of the given array.
Parameters:
items – an array of objects to locate in the ContentInfoBean.
Returns:
true if there is only one object and it is contained, false otherwise

---

containsComponent

```
public boolean containsComponent (Component  component)
```

Check if the given component is contained by this InfoBean
Parameters:
component – the Component to check for
Returns:
true if it is contained, else false

---

getLeifDataItemMenu

```
public JMenu getLeifDataItemMenu (LeifDataItem dataItem,
                                  boolean showCutPasteItems)
```

Return the data item menu for a LeifDataItem (usually the selected LeifDataItem). This is used by the menubar to add menuitems from this JMenu to a Data menu if there is exactly one DataItemSelected selected.
Overrides:
getLeifDataItemMenu in class DataItemSinkUIBean
Parameters:
dataItem – the LeifDataItem to get the menu for
showCutPasteItems – true to allow cut and paste items to appear, false to omit them.
Returns:
the JMenu for the given LeifDataItem

---

isDragEnabled

```
public boolean  isDragEnabled ()
```

Return true if the default Drag support enabled.
Overrides:
isDragEnabled in class DataItemsSinkUIBean
Returns:
true if drag is enabled, false if not, default is initialized to true.

---

setDragEnabled

```
public void setDragEnabled (boolean  enabledrag)
```

Set the status of the default Drag support.
Overrides:
setDragEnabled in class DataItemSinkUIBean
Parameters:

FIG. 33P

enabledrag – true if default drag support should be used, false if not.

---

setDragOwnerProxy

```
public void setDragOwnerProxy (DragOwner  dragProxy)
```

Set a DragOwner "proxy" for this InfoBean.
Overrides:
setDragOwnerProxy in class DataItemSinkUIBean
Parameters:
dragProxy – a DragOwner implementation.

---

isDropEnabled

```
public boolean isDropEnabled ()
```

Return true if the default Drop support is enabled.
Overrides:
isDropEnabled in class DataItemSinkUIBean
Returns:
true if drag is enabled, false if not, default is initialized to true.

---

setDropEnabled

```
public void setDropEnabled (boolean  enabledrop)
```

Set the status of the default Drop support.
Overrides:
setDropEnabled in class DataItemSinkUIBean
Parameters:
enabledrop – true if default drag support should be used, false if not.

---

setDropOwnerProxy

```
public void setDropOwnerProxy (DropOwner  dropProxy)
```

Set a DropOwner "proxy" for this InfoBean.
Overrides:
setDropOwnerProxy in class DataItemSinkUIBean
Parameters:
dropProxy – a DropOwner implementation.

Package com.xis.leif.im

This package contains classes that provides the APIs for using information management in applications.

See:

Description

Interface Summary	
Attribute	The Attribute class represents an attribute for a particular data type.
AttributeAlias	The AttributeAlias indicates an alias from 1..n Attributes to a single Attribute, together with a precision level; the higher the precision, the better the alias.
AttributeFactory	The AttributeFactory class allows an implementor to return an appropriate Attribute for the given LeifDataItem.
AttributeLookup	The AttributeLookup interface is used to lookup Attribute objects for a particular data item.
DisplayLabel	The DisplayLabel Interface defines methods that are needed for use with DisplayLabelAttributes.
Domain	This interface describes the basic fields and methods possessed by all Domains.
InfoModel	The InfoModel interface is the interface that is used to convert raw data items into LeifDataItems.
LeifDataItem	The LeifDataItem interface represents a simple data item.
LeifDataItemObserver	This class is used for observing a LeifDataItem to know when it has finished processing an action.
LiteDataItem	The LiteDataItem interface represents a data item.
PropertyProvider	If a PropertyProvider implementation is added to services it can be used to replace the standard behavior when a PropertySheetView is opened from a JAF menu or as a default command.
RawDataItemLookup	The RawDataItemLookup interface is used to look up a raw data item from a unique id.

Class Summary	
AbstractAttribute	The AbstractAttribute class represents an Attribute.
AttributeAliasPluggableService	This register AttributeAliases.
AttributeDescriptor	The AttributeDescriptor class is used to describe an attribute without providing functionality of how to use the attribute.
AttributeDescriptorFactory	The AttributeDescriptorFactory class is a singleton class used to create or get AttributeDescriptors.
AttributeFactoryInfoModelSubset	The AttributeFactoryInfoModelSubset class provides an InfoModel that will add the Attributes specified by the AttributeFactories to all LeifDataItems this InfoModel creates.

FIG. 34A

FIG. 34B

AttributeGetRequest	The AttributeGetRequest class is used to package all the necessary parameters for getting the value an attribute.
AttributeLockRequest	The AttributeLockRequest class bundles attributes needed to gain access to locked LeifDataItems.
Attributes	The Attributes is a container for holding attributes.
AttributeSetRequest	The AttributeSetRequest class is used to package all the necessary parameters for setting an attribute.
BaseDataItem	The BaseDataItem is the first wrapper around raw data items.
BaseInfoModel	//PENDING(RK): Any method marked with "PENDING" in the JavaDoc will//likely be removed before XIS is released in final form.
BaseInfoModelServicesProvider	The BaseInfoModelServicesProvider class will provide all the services available from the BaseInfoModel to the given BeanContextServices object.
Collection_Properties	This class populates JAFMenus for generic collections.
DataItemActionManager	The DataItemActionManager class provides some useful static methods for dealing with actions on LeifDataItems.
DataItemActionManager.LeifReferenceActionListener	This class is an actionListener to be used with LeifReference "Load" menus.
DataItemMenuSet	The DataItemMenuSet class is used by the LeifJAFUtilities class to return essentially a DataItem popup menu with annotation.
DataItemMenuSetEntry	The DataItemMenuSet.Entry class encapsulates a DataItem and it's menu, and also provides some convenience methods for adding additional menu items.
DefaultWrapperAttribute	The DefaultWrapperAttribute class is a generic attribute that is the superclass of all defaults in generated domain attributes.
DisplayLabelAttribute	The DisplayLabelAttribute class is used to display one or more Attribute values in conjunction with string literals specified by users.
DisplayLabelData	The DisplayLabelData class is used by the DisplayLabelAttribute to store a mapping of LeifDataItems to DisplayLabelTemplates.
DisplayLabelTemplate	The DisplayLabelTemplate class is used by the DisplayLabelAttribute to compute and store editing and rendering values for every LeifDataItem that has the attribute.
DomainMethod	Abstractly represents a Domain Method.
DomainMethodDescriptor	The DomainMethodDescriptor class is used to describe a DomainMethod.
DomainMethodDescriptorFactory	The DomainMethodDescriptorFactory class is a singleton class used to create or get DomainMethodDescriptors.
DomainWrapper	This class adds methods to LeifDataItem delegator that are useful in the domain wrappers.
DynamicAttributes	The DynamicAttributes class is used for storing dynamic attributes.

FIG. 34C

FieldMetaData	FieldMetaData specifies sorting and subset criteria for an attribute.
FieldMetaDatas	The FieldMetaDatas class represents a collection of FieldMetaData for a data item.
InfoModelDataItem	The InfoModelDataItem allows views to wrap LeifDataItems and add/remove/modify attributes that will only affect that view.
InfoModelSubset	Typically when creating an InfoModel to nest within an existing InfoModel, which is done by ViewUIBeans, an InfoModelSubset is used.
InvalidWrapperAttribute	The InvalidWrapperAttribute class
LeifDataItemComparator	The LeifDataItemComparator compares LeifDataItems by AttributeDescriptor supplied by the user.
LeifDataItemDelegator	Implements the methods in LeifDataItem in a wrapper so you don't have to.
LeifDataItemSorter	The LeifDataItemSorter provides a default sorting tool for all LEIF LeifDataItem objects.
LeifDataItemUpdate	This classes is used with the LeifDataItemObserver.
LeifInitialization	The LeifInitialization class handles some standard initialization for most XIS Applications.
LeifJAFUtilities	The LeifJAFUtilities class provides some useful static methods for Leif–related JavaBeans Activation Framework (JAF) processing.
LeifJAFUtilities.LeifReferenceActionListener	This class is an actionListener to be used with LeifReference "Load" menus.
LeifRequest	The LeifRequest class is used to package all the necessary parameters for requesting information for a LeifDataItem.
LeifTransaction	The LeifTransaction class is used to construct a transaction.
LockedLeifDataItem	The LockedLeifDataItem class is used to enable locking on the data item.
MethodRequest	The MethodRequest class is used to package all the necessary parameters for invoking a DomainMethod for a LeifDataItem.
MutableAttributeDescriptor	Mutable subclass of AttributeDescriptor.
ObserverSupport	This class provides useful support for using the LeifDataItemObserver.
RequestPool	The RequestPool class is used to to assist Object pooling.
Resources	The Resources class is automatically generated and must be public, but it is intended to be used only by Java's internationalization support classes.
SelectableDataItem	Creates a wrapper around a LeifDataItem for a SelectableInfoModel.
SelectableInfoModel	Manages selections for the selectable leif data items that are contained within this model.
Translator	A major design goal for XIS was to provide the ability to integrate existing data item classes without modifying them.

FIG. 34D

TranslatorRegistry	Provides a central location for maintaining Translators, extended Translators and locating Domain methods on data items.
UndefinedAttribute	The UndefinedAttribute class represents an undefined attributes.
VisibilityAttribute	The VisibilityAttribute class is an Attribute that is ready to use for LeifDataItem visibility.

Exception Summary	
DataItemNotFoundException	The DataItemNotFoundException class is an exception that can be thrown when trying to look up a data item from an id
InvalidObjectSchemaException	Signals that there was a problem with the creation or modification of an ObjectSchema.
TranslatorException	The TranslatorException class
UnconvertibleAliasException	Indicates that the requested attribute alias could not be calculated or converted.
UndefinedLeifAttributeException	Indicates that the requested attribute is not applicable for the object.
UndefinedLeifMethodException	The UndefinedLeifMethodException class indicates that the data item does not define the method.
UnremovableAttributeException	The UnremovableAttributeException class indicates an attempt to remove an Attribute that was defined by the raw data item (either by reflection or a Translator.) Only additional Attributes added to LeifDataItems can be removed.

FIG. 35A

com.xis.leif.im

Interface InfoModel

All Superinterfaces:
BeanContextChildOwner, BeanContextChildOwnerDelegator, BeanContextProxy

All Known Implementing Classes:
InfoModelSubset

---

public interface InfoModel
extends BeanContextChildOwnerDelegator

The InfoModel interface is the interface that is used to convert raw data items into LeifDataItems. The InfoModel should hold each of these LeifDataItems created using weak references so that the data items can be cleaned up when they are no longer being used. //PENDING(RK): Any method marked with "PENDING" in the JavaDoc will likely be removed before LEIF is released in final form.

Since:
LEIF 4.0
Version:
$Revision: 1.20 $, $Date: 2001/08/17 00:54:54 $
Author:
David Almilli

Method Summary	
void	activateOneOfNService (Object service) //PENDING(RK): This method will probably be removed from InfoModel! Notify the InfoModel that the given service is the preferred service of its type, and that this particular object should be returned if its class is requested, until removed or until another object of the same type is passed to a future call to this method.
void	addInfoModelListener (InfoModelListener listener) Adds a listener to this InfoModel so that the listener will be informed of changes to the InfoModel.
void	addOneOfNService (Object service) //PENDING(RK): This method will probably be removed from InfoModel! Add an object as a service to be retrieved by a call to getService() (via BeanContext APIs) on any class that this object implements or extends.
void	clearSelection () Clears the selection.
LeifDataItem []	dump () Gives a list of all the LeifDataItems currently in the InfoModel.
EzContext	getEzContent () Gets an EZ Context that corresponds to this InfoModel so the developer can use the EZ APIs.

FIG. 35B

LeifDataItem	getLeifDataItem (long uid) This will attempt to lookup a LeifDataItem from an id.
LeifDataItem	getLeifDataItem (Object rawDataItem) This will wrap a raw java Object with a LeifDataItem wrapper so you can use it in leif as a data item.
LeifDataItem	getLeifDataItem (Object rawDataItem, boolean create) This will wrap a raw java Object with a LeifDataItem wrapper so you can use it in leif as a data item.
LeifDataItem []	getLeifDataItems (Object [] rawDataItems) This convience method will wrap an array of raw java Objects with LeifDataItem wrappers so you can use them in leif as LeifDataItems.
InfoModel	getParentInfoModel () Provides access to the parent InfoModel that this InfoModel delegates to.
Object []	getSelectedRawDataItems () Gets the list of all the currently selected items for this InfoModel
Object	getSingleSelectedItem () Get the selected raw data item, if only one.
ViewHost	getViewHost () Gets the ViewHost that this InfoModel is associated with.
void	removeInfoModelListener (InfoModelListener listener) Removes a listener from this InfoModel so that the listener will no longer be informed of changes to the InfoModel.
void	removeOneOfNService (Object service) //PENDING(RK): This method will probably be removed from InfoModel! Remove an object that was a service to be retrieved by a call to getService () (via BeanContext APIs) on any class that this object implements or extends.

Methods inherited from interface com.xis.beancontext.BeanContextChildOwnerDelegator
initializeBeanContextResources, releaseBeanContextResources

Methods inherited from interface com.xis.beans.beancontext.BeanContextChildOwner
getOwnedBeanContextChild

Methods inherited from interface java.beans.beancontext.BeanContextProxy
getOwnedBeanContextChild

Method Detail

getLeifDataItem

public LeifDataItem getLeifDataItem (long uid)
throws DataItemNotFoundException

This will attempt to lookup a LeifDataItem from an id. If the UID is invalid or there isn't a LeifDataItem that already exists with that given UID, an exception will be thrown.
Parameters:
uid – the inique id for the raw data item.
Returns:

FIG. 35C

the LeifDataItem with the given UID

---

getLeifDataItem

public LeifDataItem getLeifDataItem (Object rawDataItem)

This will wrap a raw java Object with a LeifDataItem wrapper so you can use it in leif as a data item.
Parameters:
rawDataItem – the raw data that will be wrapped. *(Note: this should not already be a LeifDataItem)*
Returns:
the wrapped data item.

---

getLeifDataItem

public LeifDataItem getLeifDataItem (Object rawDataItem,
boolean create)

This will wrap a raw java Object with a LeifDataItem wrapper so you can use it in leif as a data item.
Parameters:
rawDataItem – the raw data that will be wrapped. *(Note: this should not already be a LeifDataItem)*
create – if false and the LeifDataItem is not already in the model, don't create one and return null
Returns:
the wrapped data item, or null if "create" is false and not found

---

getLeifDataItems

public LeifDataItem [] getLeifDataItems (Object [] rawDataItems)

This convenience method will wrap an array of raw java Objects with LeifDataItem wrappers so you can use them in leif as LeifDataItems. Note that you can get an array of raw data items often from methods like getMembers(), so this is a useful method to have.
Parameters:
rawDataItems – the raw data objects that will be wrapped. *(Note: the objects should not already be LeifDataItems)*
Returns:
the corresponding wrapped data item array.

---

getEzContext

public EzContext getEzContext ()

Gets an EZ Context that corresponds to this InfoModel so the developer can use the EZ APIs.
Returns:
the ez context for this info model

---

getSingleSelectedItem

public Object getSingleSelectedItem ()

FIG. 35D

Get the selected raw data item, if only one. Else return null.
Returns:
the selected item if there is only one.

---

getParentInfoModel

```
public InfoModel  getParentInfoModel ()
```

Provides access to the parent InfoModel that this InfoModel delegates to. If there is no parent model then this will return null.
Returns:
the parent InfoModel

---

clearSelection

```
public void clearSelection ()
```

Clears the selection.

---

getSelectedRawDataItems

```
public Object []  getSelectedRawDataItems ()
```

Gets the list of all the currently selected items for this InfoModel
Returns:
all of the selected data items (as raw data items)

---

activateOneOfNService

```
public void activateOneOfNService  (Object service)
```

//PENDING(RK): This method will probably be removed from InfoModel! Notify the InfoModel that the given service is the preferred service of its type, and that this particular object should be returned if its class is requested, until removed or until another object of the same type is passed to a future call to this method.
Parameters;
service – the object to become the preferred service

---

addOneOfNService

```
public void addOneOfNService (Object service)
```

//PENDING(RK): This method will probably be removed from InfoModel! Add an object as a service to be retrieved by a call to getService() (via BeanContext APIs) on any class that this object implements or extends.
Parameters;
service – the object to be returned when requested

FIG. 35E

removeOneOfNService

```
public void  removeOneOfNService (Object  service)
```

//PENDING(RK): This method will probably be removed from InfoModel! Removed an object that was a service to be retrieved by a call to getService() (via BeanContext APIs) on any class that this object implements or extends.
Parameters:
service – the object to be removed from service

---

getViewHost

```
public ViewHost  getViewHost ()
```

Gets the ViewHost that this InfoModel is associated with. If this InfoModel is not associated with a ViewHost then this will return null.
Returns:
the view host that is maintaining this InfoModel.

---

addInfoModelListener

```
public void addInfoModelListener (InfoModelListener listener)
```

Adds a listener to this InfoModel so that the listener will be informed of changes to the InfoModel.
Parameters:
listener – the listener to add

---

removeInfoModelListener

```
public void removeInfoModelListener (InfoModelListener listener)
```

Removes a listener from this InfoModel so that the listener will no longer be informed of changes to the InfoModel.
Parameters:
listener – the listener to remove

---

dump

```
public LeifDataItem []  dump()
```

Gives a list of all the LeifDataItems currently in the InfoModel. It is highly recommended to use this method only if you absolutely have no other way of accomplishing the task you need to do. Please keep in mind that if you hold onto the LeifDataItems contained in the array returned or if you hold onto the array itself, the items will not be removed from InfoModel until you release them. If you wish to hold onto them, you should wrap them in WeakReference objects.

Note: When you use the dump() method in combination with the addInfoModelListener so that you can keep track of the same set of LeifDataItems as the InfoModel, you can synchronize on the InfoModel to get the dump and then add a listener to receive events of future changes.

Example:

```
synchronized(infoModel)  {
    LeifDataItem[] dataItems = infoModel.dump();
    infoModel.addInfoModelListener(this);
    for  (int i=0; i < dataItems.length;  i++)  (
        processItem(dataItems[i]);
    }
}
```

Returns:
the list of LeifDataItems currently in the InfoModel.
See Also:
WeakReference

FIG. 35F

FIG. 36A

Package com.xis.leif.event

This package contains classes for handling events in XIS.

See:
Description

Interface Summary	
InfoModelListener	The InfoModelListener is used to monitor changes to an InfoModel.
LeifDataItemListener	This class is used for listening to LeifDataItems for various events.

Class Summary	
AttributeChangedEvent	An " AttributeChanged " event gets delivered whenever a data item changes an attribute value.
ContainerAddedEvent	A " ContainerAdded " event gets delivered whenever a data item is contained as a member in a new object.
ContainerRemovedEvent	A " ContainerRemoved " event gets delivered whenever a data item has been removed as a member from a containing object.
DataItemReplacedEvent	The DataItemReplacedEvent class is used to indicate member changes of a containing data item.
InfoModelEvent	The InfoModelEvent gets delivered whenever a LeifDataItem is created by the InfoModel, or when a LeifDataItem has been " lost " by the InfoModel.
InfoModelEvenSupport	The InfoModelEvenSupport support class provides basic support for managing listeners on an InfoModel.
LeifDataItemAdapter	The LeifDataItemAdapter class provides support for setting up a LeifDataItemListener on a data item.
LeifEvenSupport	This is a utility class for XIS developers to use when they want to fire event changes.
MemberAddedEvent	The MemberAddedEvent class indicates that members were added to this data item.
MemberEvent	The MemberEvent class is used to indicate members changes of a containing data item.
MemberRemovedEvent	The MemberRemovedEvent class indicates members that the members are being removed from the containing data item.
ReferenceAddedEvent	The ReferenceAddedEvent class indicates that LeifReferences were added to the LeifDataItem.
ReferenceEvent	The ReferenceEvent class indicates changes to the LeifReferences of the data item.

ReferenceRemovedEvent	The ReferenceRemovalEvent class indicates that LeifReferences were removed from the LeifDataItem.
ReferrerAddedEvent	The ReferrerAddedEvent class indicates that a Referrer was added to the data item.
ReferrerRemovedEvent	The ReferrerRemovedEvent calss indicates that a Referrer was removed from the data item.

Package com.xis.leif.event Description

This package contains classes for handling events in XIS.

FIG. 36B

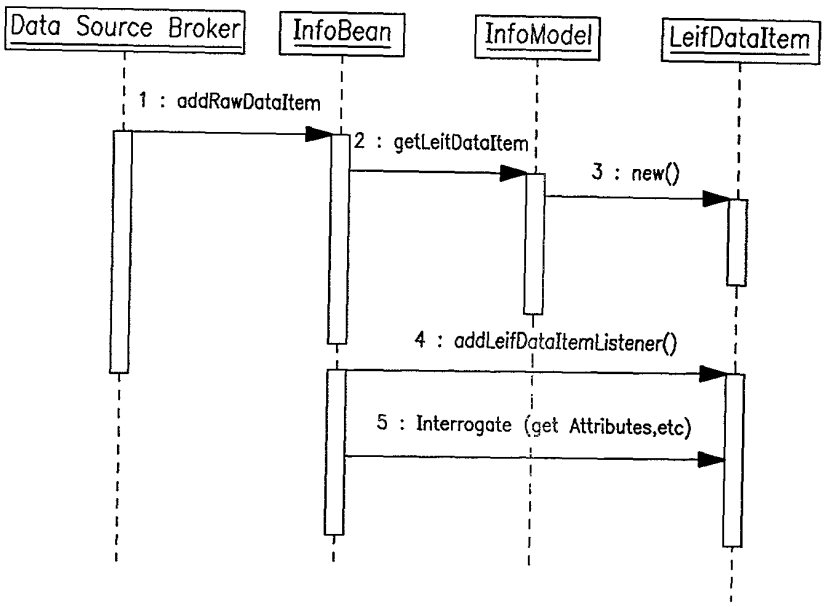

Data Source Broker
InfoBean
InfoModel
LeifDataItem
1 : addRawDataItem
2 : getLeifDataItem
3 : new()
4 : addLeifDataItemListener()
5 : Interrogate (get Attributes,etc)

FIG. 36C

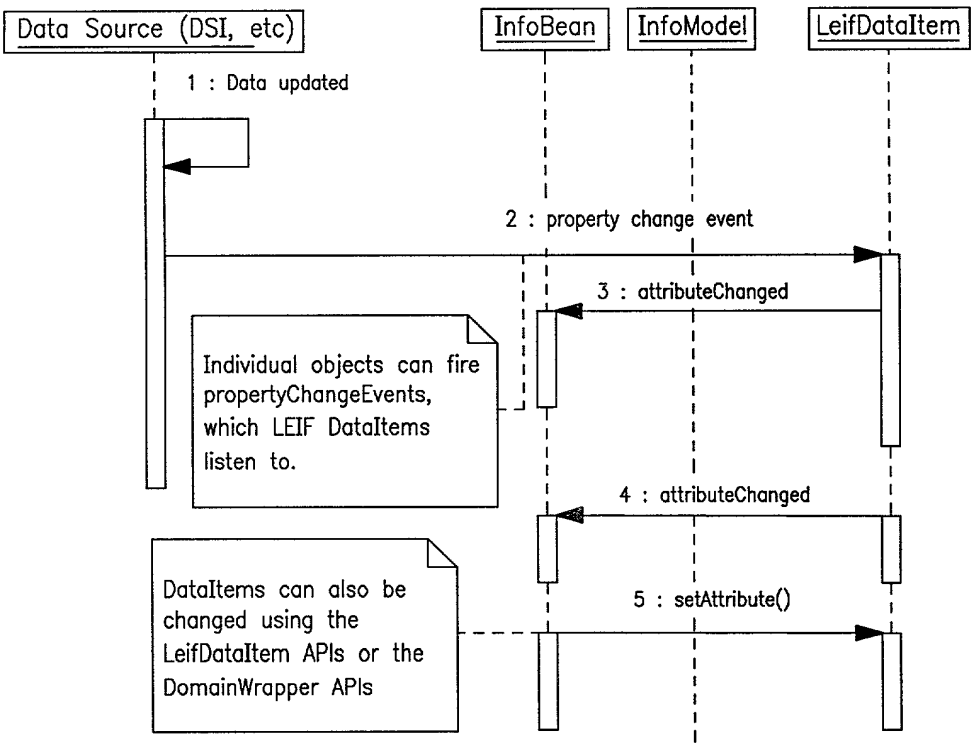

Data Source (DSI, etc)
InfoBean
InfoModel
LeifDataItem
1 : Data updated
2 : property change event
3 : attributeChanged
Individual objects can fire propertyChangeEvents, which LEIF DataItems listen to.
4 : attributeChanged
DataItems can also be changed using the LeifDataItem APIs or the DomainWrapper APIs
5 : setAttribute()

FIG. 36D

TestHarness.java

```
/* XIS Tutorial standalone sequence example 5 XIS interfacing. */

import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
/*{*/
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.Dimension;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import javax.swing.JSplitPane;
import javax.swing.JComponent;
/*}*/
import jclass.chart.JCChart;
import com.xis.leif.im.BaseInfoModel;
import com.xis.plot.PlotInfoBean;
import com.xis.plot.chartviews.LeifChartView;
/*{*/
import com.xis.table.TableInfoBean;
import com.xis.tree.TreeInfoBean;
/*}*/

public class TestHarness {

   public static void main(String[] args) {

      // the plugin manager is only required for more complex applications
      // involving multiple components integrated at runtime
      BaseInfoModel.setStartingPlugInManager(false);

/*{*/
      HelloWorld hello1 = new HelloWorld("First HelloWorld object.");
      HelloWorld hello2 = new HelloWorld("Second HelloWorld object.");
      HelloWorld hello3 = new HelloWorld("Third HelloWorld object.");
      HelloWorld hello4 = new HelloWorld("Fourth HelloWorld object.");
      HelloWorld hello5 = new HelloWorld("Fifth HelloWorld object.");
```

FIG. 37A

Continuation of TestHarness.java

```
        Object[] helloArray = new Object[] { hello1, hello2, hello3, hello4,
                                hello5 };

        // create table and plot infobeans to display HelloWorld objects
        TableInfoBean table = new TableInfoBean();
/*}*/
        PlotInfoBean plot = new PlotInfoBean();
        plot.setChartType(JCChart.BAR);
        // the alternatives are SCATTER_PLOT, PLOT, AREA, PIE, CANDLE,
        // and STACKING_BAR, though not all will make sense in this example

        // We can set the attribute for initial display on the plot;
        // see step 3 for further comments.
        plot.setYAxisAttribute(
           "com.xis.domains.movement.MovementDomain.speed");
        plot.setDynamicAdjustment(true);  // so axes track value magnitude
        plot.setBarChartAdjusting(true);  // needed in some cases for bar chart

/*{*/
        // a top-level frame as before to hold both plot and table side-by-side
        JFrame tablePlotFrame = new JFrame("HelloWorld(s) Table/Plot");
        // use a repaired JSplitPane (see below) to manage the two beans
        SaneJSplitPane splitpane = new SaneJSplitPane(table,plot,
           new Dimension(table.getPreferredSize().width +
                    plot.getPreferredSize().width,
                    Math.min(table.getPreferredSize().height,
                         plot.getPreferredSize().width)),
           0.50);
        tablePlotFrame.getContentPane().add(splitpane);
        tablePlotFrame.pack();
        tablePlotFrame.setVisible(true);

        // a tree infobean to display our HelloWorld objects
        TreeInfoBean tree = new TreeInfoBean("HelloWorld(s) Tree");
        tree.addRawDataItems(helloArray);

        // a top-level frame to hold our tree infobean
        JFrame treeFrame = new JFrame("HelloWorld(s) Tree");
```

FIG. 37B

Continuation of TestHarness.java

```
        // avoid placing the windows on top of one another if we can
        int cutoffHeight = 424;
        if (Toolkit.getDefaultToolkit().getScreenSize().getHeight() >
            cutoffHeight + 200) {
            treeFrame.setLocation(348,cutoffHeight+7);
        }
/*}*/
        // add a listener for window closing
        treeFrame.addWindowListener(
            new WindowAdapter() {
                public void windowClosing(WindowEvent e) {
                    System.exit(0);
                }
            }
        );

        // stick the tree infobean in the frame and display it
        treeFrame.getContentPane().add(tree);
        treeFrame.pack();
        treeFrame.setVisible(true);

    } // main


/*{*/
    /**
     * This class overrides the default JSplitPane to provide a reasonable
     * resize behavior: maintain the left and right panels in the same
     * proportions.
     */
    public static final class SaneJSplitPane extends JSplitPane {

        private int    lastWidth;
        private double lastDividerProp;

        public SaneJSplitPane(JComponent leftComponent,
                              JComponent rightComponent,
                              Dimension dims, double startProportion) {
```

FIG. 37C

Continuation of TestHarness.java

```
        super(JSplitPane.HORIZONTAL_SPLIT,
            leftComponent, rightComponent);
        setSize(dims);

        // Since the JSplitPane doesn't set the lastDividerLocation
        // variable, nor does it provide any other easier way to maintain
        // the split proportion on resize, we must track the divider
        // location ourself.
        lastWidth = dims.width;
        lastDividerProp = startProportion;
        setDividerLocation(startProportion);

        // this listens for resize events on the splitpane and makes sure
        // we keep same split proportions
        addComponentListener(new ComponentAdapter() {
            public void componentResized(ComponentEvent event) {
                setDividerLocation(lastDividerProp);
                lastWidth = (int)event.getComponent().
                    getSize().getWidth();
            } });
        // only way to know if divider moved by user is to listen for
        // resize events on the components; this isn't foolproof (since
        // resizes can come from other sources) but it works well enough
        leftComponent.addComponentListener(new ComponentAdapter() {
            public void componentResized(ComponentEvent event) {
                // we add in getDividerSize() / 4 to compensate for a
                // bug in JSplitPane which doesn't take account of the
                // divider width in location-proportion conversions
                lastDividerProp = (double)(getDividerLocation() +
                    (getDividerSize() / 4)) / lastWidth;
            } });

    }

  }
/*}*/

}
```

FIG. 37D

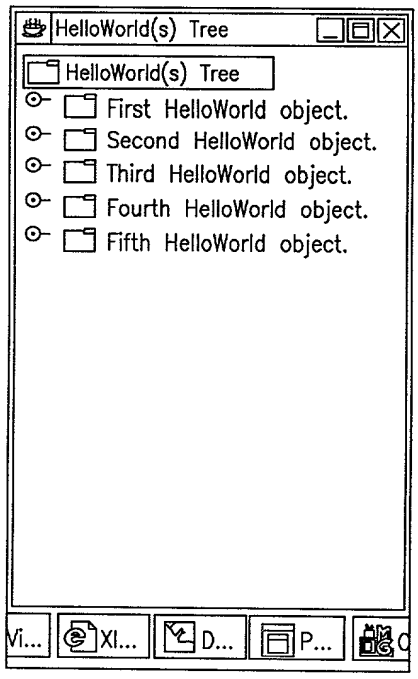


FIG. 38A

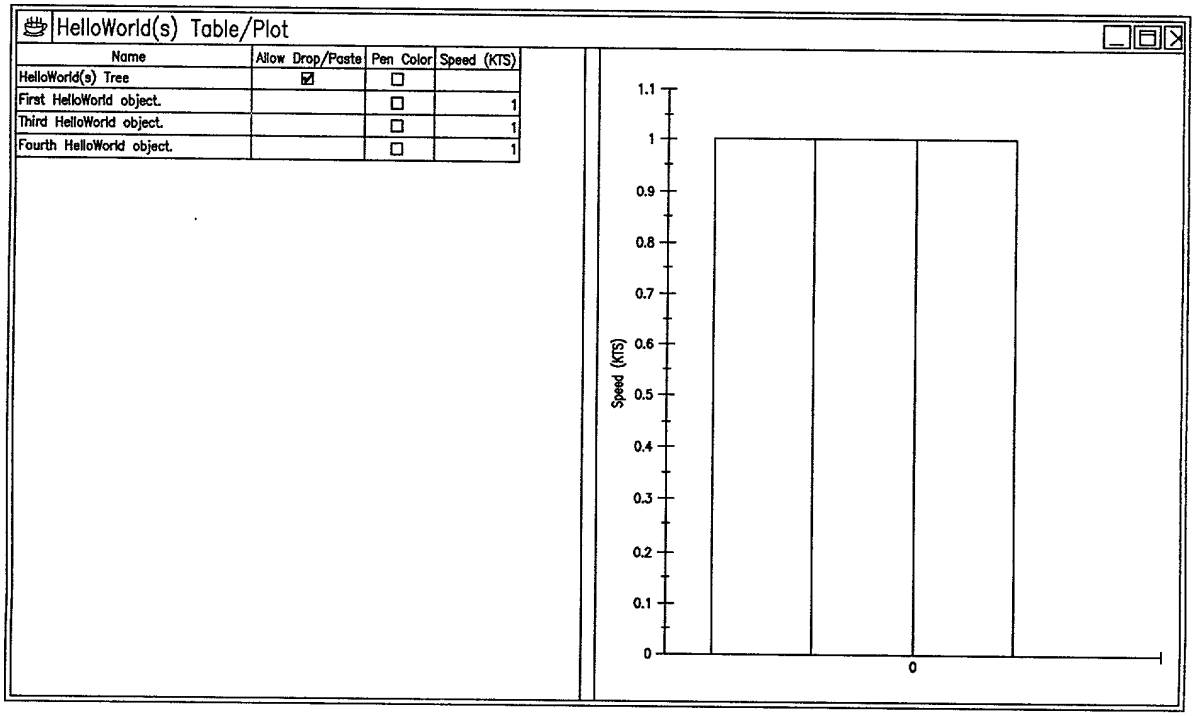


FIG. 38B

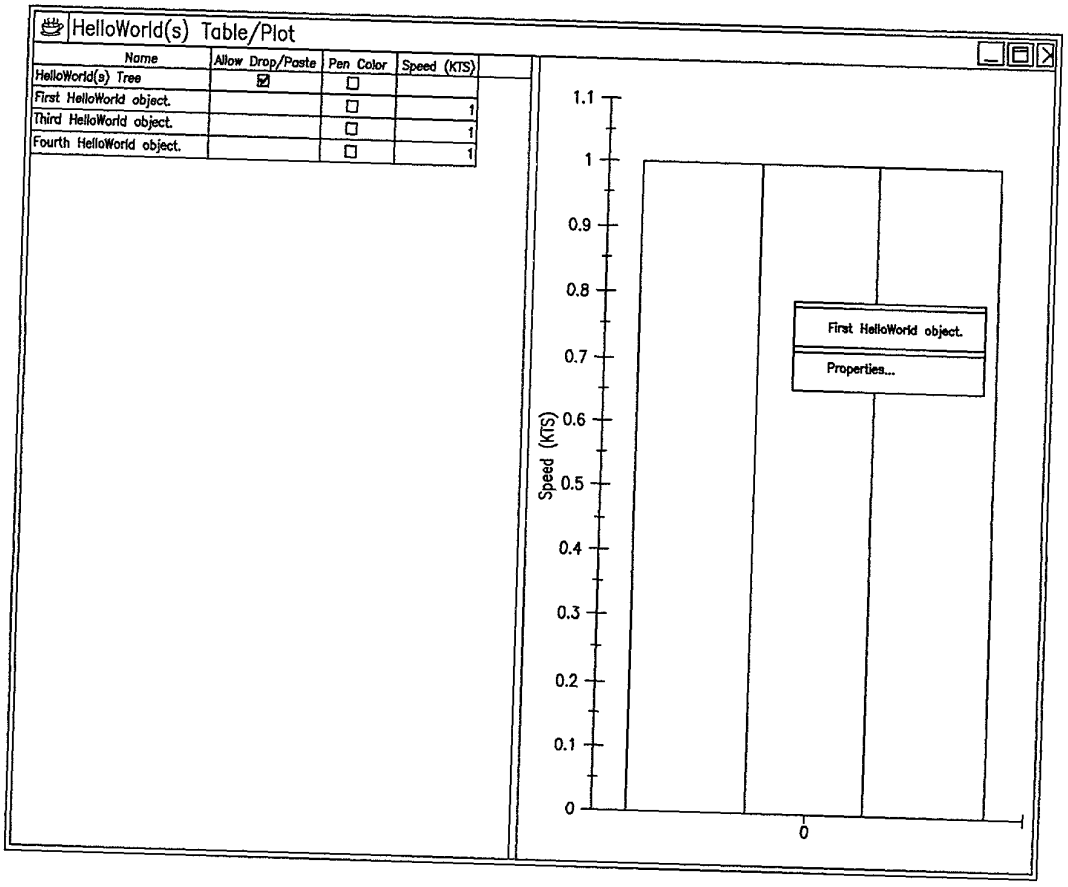

HelloWorld(s) Table/Plot
Name
Allow Drop/Paste
Pen Color
Speed (KTS)
HelloWorld(s) Tree
First HelloWorld object.
Third HelloWorld object.
Fourth HelloWorld object.
1
1
1
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Speed (KTS)
First HelloWorld object.
Properties...
0

FIG. 38C

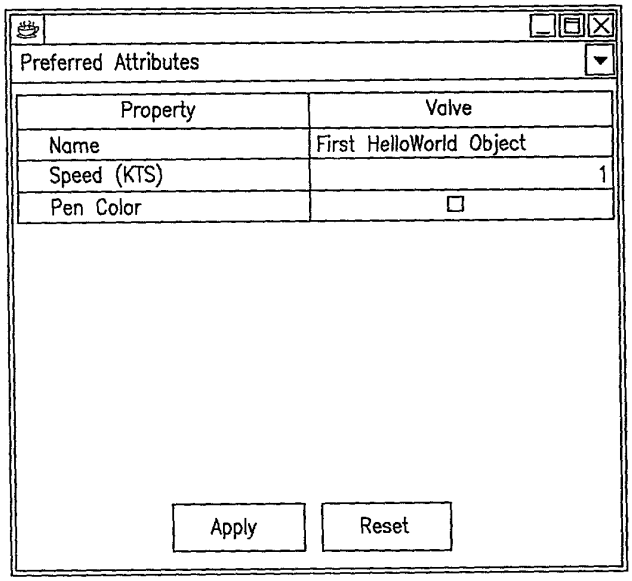

Preferred Attributes
Property
Valve
Name
First HelloWorld Object
Speed (KTS)
1
Pen Color
Apply
Reset

FIG. 38D

FIG. 39A

comm.xis.leif.im

Interface AttributeAlias

public interface AttributeAlias

The AttributeAlias indicates an alias from 1..n Attributes to a single Attribute, together with a precision level; the higher the precision, the better the alias. The alias allows a caller to query for one of the 1..n "from" attributes and get the value stored by the data item under the "to" attribute, possibly mediated by some conversion, such as a units transformation.

If the converted or calcuated value cannot be determined, then the Attribute#getValue () method should throw an UnconvertibleAliasException. This Exception is a subclass of the UndefinedLeifAttributeException which is typically thrown by normal Attributes in this case, and it can provide a descriptive message indicating the source of the incompatibility.

The utility of attribute aliases may be seen by considering the following example:

The user has performed a query from an external data source and retrieved a set of Airfields, indexed by an ICAO identifier. The user now wants to get the list of Aircraft at one of the airfields. There is a local aircraft database with a foreign key for Airfields, but that key is a WAC identifier, not ICAO.

Assumption: The application was NOT written ahead of time to know about these two databases or their ID types. Instead, what you have is an XIS "LeifDataItem" for the Airfield, and you have an XIS InfoBean for the Aircraft query form.

What you want to do is to copy (or "drop") the Airfield data item into the "WAC" field in the query form. In doing this, the Form will ask the data item for its "WAC" attribute (because this is all it knows about). It uses the "getWAC ()" method from some domain (say, the AirfieldDomain).

The way this could work is that there would have to be an AttributeAlias defined to convert ICAO to WAC – or, more specifically, AviationDomain. ICAO to AirfieldDomain.WAC. The AttributeAlias returns an attribute object that knows how to transform ICAOs to WACs (e.g., by accessing a conversion table). The Attribute, in turn, has a getValue () method to execute that transformation and return the WAC.

This process would be entirely transparent to the user, or even the caller, who would just see a result returned from the getWAC () method. In cases where the conversion was not possible, the UnconvertibleAliasException would be thrown, possibly providing informative information to the caller or user.

Finally, note that due to the way the mechanism is set up (using resources and a PluggableService), this AttributesAlias can be installed as a separate module without requiring any re–coding or re–compliation of the existing application.

FIG. 39B

Method Summary	
AttributeDescriptor []	getAliasedFrom () This indicates which AttributeDescriptors (which in turns means which Attributes) are required for the alias.
AttributeDescriptor	getAliasedTo () This indicates which AttributeDescriptor that this AttributeAlias is for.
Attribute	getAttribute () Get the Attribute object that is the alias Attribute.
int	getPrecisionPriority () This indicates the precision of the AttributeAlias.

Method Detail

getPrecisionPriority

public int getPrecisionPriority ()

This indicates the precision of the AttributeAlias. The higher the number the better the alias. This number is used to determine which AttributeAlias to use when there are more than one alias for a given Attribute.

Returns:
the precision of the alias.

getAliasedFrom

public AttributeDescriptor [] getAliasedFrom ()

This indicates which AttributeDescriptors (which in turns means which Attributes) are required for the alias.

Returns:
the list of descriptors required for this alias.

getAliasedTo

public AttributeDescriptor getAliasedTo ()

This indicates which AttributeDescriptor that this AttributeAlias is for.

Returns:
the descriptor that this alias is for.

getAttribute

public Attribute getAttribute()

Get the Attribute object that is the alias Attribute. This attribute is responsible for performing the translation from the aliased from Attributes. The AttributeDescriptor of the Attribute MUST be the same AttributeDescriptor returned by getAliasedTo.

Returns:

the attribute that will do the translation.

FIG. 39C

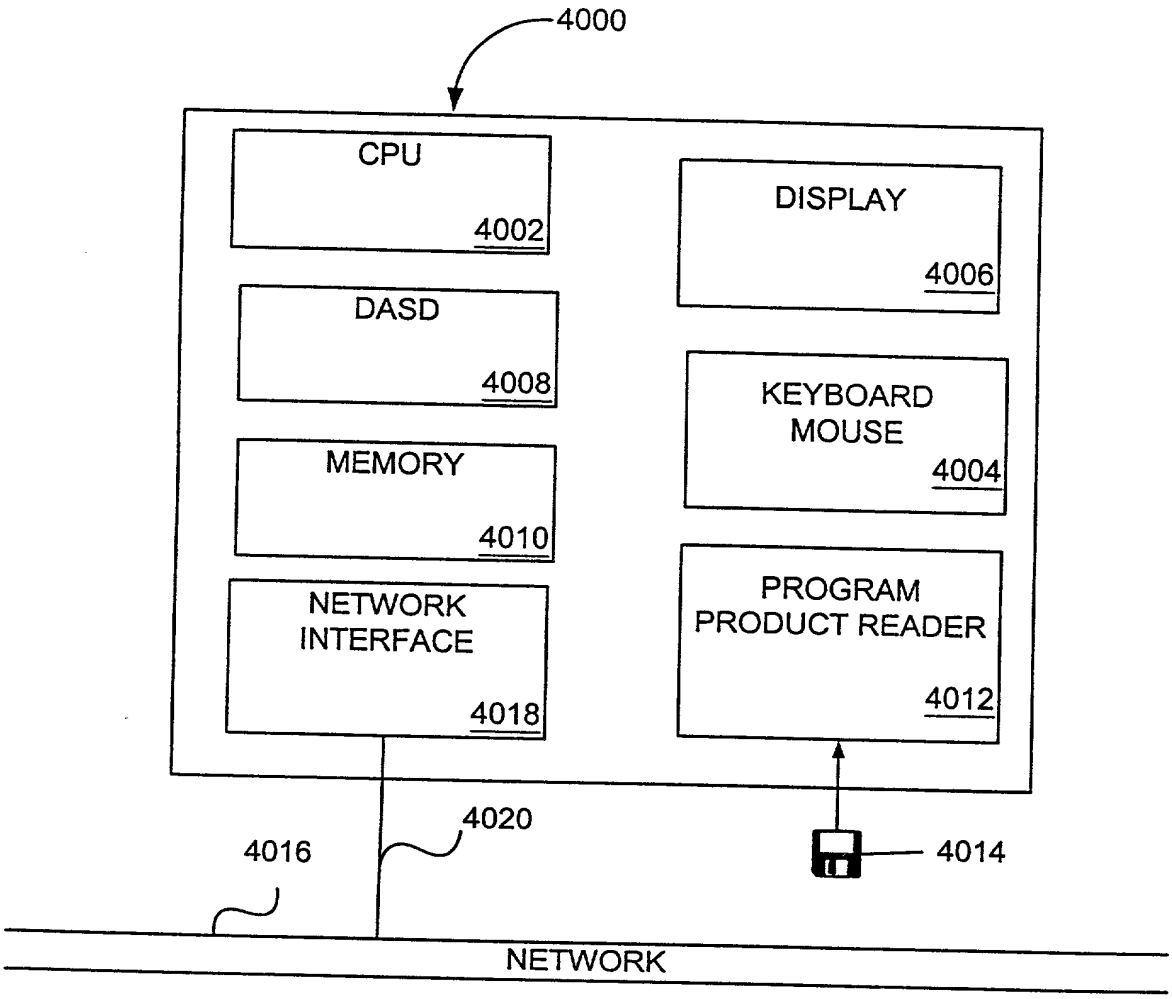

4000
CPU
4002
DISPLAY
4006
DASD
4008
KEYBOARD
MOUSE
4004
MEMORY
4010
NETWORK
INTERFACE
4018
PROGRAM
PRODUCT READER
4012
4020
4016
4014
NETWORK

FIG. 40